Unifying Strategies for Web Augmentation

Niels Olof Bouvin

Aarhus University,
Department of Computer Science,
Aabogade 34A, DK8200 Aarhus N, Denmark

bouvin@daimi.aau.dk

ABSTRACT

Since the beginning of the WWW, tools have been developed to augment the functionality of the Web. This paper provides an investigation of hypermedia tools and systems integrating the World Wide Web with focus on functionality and the techniques used to achieve this functionality. Similarities are found and based on this, a new framework, the Arakne framework, for developing and thinking about Web augmentation is presented. The Arakne framework is flexible and supports most kinds of Web augmentation. Finally an implementation of the Arakne framework is described and discussed.

KEYWORDS

Web Integration, Open Hypermedia Systems, Open Hypermedia Protocol, Collaboration on the Web, Unifying interfaces, Common Reference Architecture for open hypermedia systems, Java

INTRODUCTION

The World Wide Web has in an amazing short time span become the hitherto largest hypertext and is pervasive in everyday life as few things before. This success has in part been attributed to the simple architecture behind the Web: A stateless file transfer protocol (HTTP), an universal Internet naming scheme (URL) and an easily understood document format (HTML). These standards have (largely) been adhered to, and this has enabled the creation of a large amount of software, be it Web servers or browsers to work together to the benefit of all.

The success of this simple and hugely scaleable architecture has come, from the standpoint of the hypermedia research community, at some costs, as the Web itself is lacking in the ways of more advanced (but ironically often far older) hypermedia systems. Web links are unidirectional jump links, embedded in the HTML documents, severely diminishing the flexibility of use. There is yet no widespread support for collaborative authoring, though an initiative such as WebDAV [13]) holds great promise.

An important development in the hypermedia community in the nineties has been the focus on and development of open hypermedia systems integrating third-party applications. Significant work has been done in systems such as MicroCosm [9][20], HyperDisco [44][45], HOSS [32], DHM [14][15], and Chimera [1]. These systems have all addressed the problem of augmenting third-party applications, and the lessons learned are important guidelines for future work of Web augmentation. Based on these experiences,, taxonomies and studies have been developed, by Whitehead [43], Grønbæk & Wiil [17], and Wiil & Østerbye [46][48], to help researchers and developers discuss and reason about the open hypermedia field and how to utilise hypermedia in third-party applications. Given this experience of integration it should come as no surprise that the open hypermedia community was quick to develop open hypermedia Web integrations, e.g. DLS [7], DHM/WWW [16], and Chimera [2]. This article will investigate how they and others achieved their goals of Web augmentation.

Adhering to standards has a large part of the success of the Web and the very size of the Web has enormous inertia, so an attempt to replace the Web with something perhaps more advanced in certain aspects is, if not doomed, then up against tremendous odds. Clearly this is not the way to improve the Web. An approach that retains the benefits of the Web as well as adding new desired functionality is the development of systems that operate within Web standards, be it HTTP, HTML, browsers, or servers. This is in spirit with Meyrowitz' call for hypermedia integration in third-party applications [27], and the amount and quality of work done using this approach would suggest that it is at the very least possible.

AN OVERVIEW OF WEB AUGMENTATION STRATEGIES

As the scope of this article is to study the techniques used in Web augmentation tools, an overview of augmentation approaches is in order. This overview will work at two levels of abstraction: first a broad grouping based on the functionality of the tools and later a more specific characterisation of the individual tool. This characterisation will be based on the chosen approach to common problems, such as storage, Web browser integration, level of support for collaboration, and so forth. The former level of abstraction will let us discuss and compare related tools, while the latter will allow us to recognise reoccurring themes in the approaches taken.

A tool shall be considered a Web hypermedia augmentation tool, if it through integration with a Web browser, a HTTP proxy or a Web server adds content or controls not contained within the Web pages themselves to the effect of allowing structure to be added to the Web page directly or indirectly, or to navigate such structure. The purpose of such

Hypertext 99 Darmstadt Germany

a tool is help users organise, associate, or structure information found on the Web. This activity may be done by a single user or in collaboration with others.

Following this definition, the purpose of the Web augmentations reviewed for this article is to help users structuralise their Web work. Either by adding structure and displaying it, or by extracting structure already present and making it more visible. The displayed structure may be malleable, allowing the user to modify it. The Web augmentation tools reviewed for this article have been divided into four categories:

- Annotations/Discussion support
- Link creation and transversal
- Guided tours
- Structuring/Spatial

We claim no universality to this categorisation, but have found it handy when discussing the Web augmentation tools and their use.

A Web augmentation tool can be classified by the schema summarised in Table 1. Thus a tool can either be a part of a Web browser; it can be an closely integrated tool loaded on runtime; it can reside temporarily within the browser as an applet or an ActiveX control; it can be an application running the user's computer; or it can be located elsewhere (in that case more often than not as a HTTP server or proxy).

The Web augmentation tool can either have no storage requirements; it can store it data locally on the host computer, or remotely on a server.

Many of these Web augmentation tools modify Web pages, either to insert interfaces of their own or to add structure (e.g. links) to the Web page. This modification can take place at the Web server (perhaps a Web server translating a proprietary data format into HTML), by calling CGI-scripts that return modified pages, by using a special proxy, or by modifying the Web pages as or after they are displayed in the Web browser.

As for the collaborative aspect, a Web augmentation tool can either be strictly personal, i.e. relevant to a single user only; the created data or structures can be shared (e.g. send to another user or placed on a Web site); the structures can browsed or edited by turn-taking (asynchronously); or users can collaborate through the structures in real time (synchronously).

WEB AUGMENTATION TOOLS

In this section we will describe various Web augmentation tools focusing on the elements introduced by Table 1. The scope of this article does not allow for a comprehensive study nor a general survey, so only a subset of the existing Web augmentation tools is presented.

Annotations/Discussion support

Bush envisioned marginalia in the Memex [5], and the interest in annotations and how they should be supported by hypermedia has not diminished over the years, as witnessed by the investigation done by Marshall [25].

The first widely successful Web browser, NCSA Mosaic [30], gave the user the opportunity to create annotations to Web pages. The annotations were personal and stored locally. Later, this feature fell out of favour with Web browser developers.

Recognising that annotations whilst useful for the individual are even more beneficial for a community, several collaborative annotation tools have been created. Röscheisen *et al.*[35][36] have developed ComMentor, which have used for several purposes, including content rating and annotations. The system employs a Mosaic [30] browser, modified to provide an interface to the annotation server, which consists of a collection of CGI-scripts. The user has alongside with the browser a merge library, which inserts comments and links to comments into the Web pages. Annotations are stored in sets, of which the user may activate an arbitrary

Method of integration
A part of the browser
Browser add-ons
Within the browser (plug-ins, applets and JavaScript)
Without the browser, local
Without the browser, remote
Within the proxy
Within the Web server
Location of storage
No storage
Local storage
Remote storage
Web page modification
No modification
At the Web server
CGI-scripts at a Web server
At the proxy
In the browser
Level of collaboration supported
Personal
Shareable
Asynchronous collaborative
Synchronous collaborative

Table 1 - A classification scheme of Web augmentation tools

number. Collaborative annotation is supported through dividing users into groups that may share sets of annotations. A set of annotations can be set to be private, available to a group, or publicly available. Annotations are displayed using in-place markers (small pictures) indicating either the nature of the annotation or the author's identity. An annotation while write-protected may also be annotated.

Another system is CritLink Mediator, part of CritSuite [10], which is specialised to provide support for 'critical discussions'. CritLink employs predefined typed links (support, issue, comment, query) akin to many hypermedia systems, such as IBIS [34] and TEXTNET [40]. The comments are created by a mix of CGI-scripts, Web forms, and JavaScript and stored either on a designated server or in the user's own Web space as ordinary Web pages. Links to the comments

are inserted into Web pages by use of a Web server effectively acting as a proxy server. The pages are also modified to include a tool bar used for navigation, annotation, and the launch of CritMap [39], a tool that generates maps of neighbourhood Web pages. Links in pages presented by the server are modified to go through the server. As comments are Web pages in their own right, they can also be commented on. There are only one set of annotations and no notion of groups, though the individual user is identified.

Creating links

As noted in the introduction, quite a few research projects have addressed the issue of adding external structures to Web pages. An excellent investigation of the various approaches taken by the open hypermedia community can be found in [2]. There are two main approaches: links (or other kinds of structural information) are either displayed alongside the Web page or inserted into the Web page. The former case requires either a program to display this structure or a browser window where the structure has been converted to HTML. The latter case involves modifying HTML pages on the fly, which can be done at three places: at the origin, in transit or at arrival, i.e. the Web server, the HTTP proxy, or the Web browser.

Chimera [1][2] is an example of a system, where experiments with either displaying structure information in a separate program (an applet) or making the structure server accessible through HTTP have been carried out. By modifying a Web server to interpret HTTP requests as requests to a Chimera server to which the Web server is hooked up and in turn translating the Chimera structures to HTML, a user is able to browse the hypermedia structure using an ordinary Web browser. This experiment was extended upon by the creation of a Java applet capable of displaying Chimera hypermedia structures. By the combination of a special Web server, CGI-scripts and cookies, this applet was inserted into all pages displayed in the Web browser, giving the user immediate access to Chimera services.

Hyper-G [26] is a more specialised system, as it to achieve full functionality relies on a special document format (HTF – Hypertext Format), a special server, and a custom browser. It is however possible to interface to the system using an ordinary Web browser using a special WWW-gateway, that will translates HTF document and hypermedia structure to HTML. The hypermedia system offers strong support for hierarchical structures and searching, and allows users without a special browser to create links using forms. In recent versions HyperWave [22] (as the system is now known) offers an advanced interface utilising Java-applets and JavaScript inserted into Web pages by the HyperWave server.

DLS [6] (Distributed Link Service) is based on the MicroCosm hypermedia system [7][9][20]. The first DLS systems used a wrapper to attach a link service menu to a browser (this integration being dependent on whether an integration existed for the user's browser), thus creating a (in the terminology of Whitehead in [43]) shim integration with a third party application. Links were followed by selection of text and selecting 'Follow Link' in the attached menu. This would cause the wrapper to contact the link server with an URL encoding the request, resulting in a Web page of the matching links. To address the problem of having to install special software and to make links more visible, an interfaceless version was developed that used a link server proxy to insert links in Web pages as requested by the user. The user used a form to configure which link bases to use and how the link should be presented in the document (to make the distinction between links belonging in the document and inserted links clear). Due to performance issues (beyond 'conventional' links, MicroCosm offers computation intensive links, such as keyword links, person links and citation links) and copyright and authors' rights concerns about adding content to Web pages, a new design was introduced with the AgentDLS [8]. Rather than offering synchronous links (presented together and simultaneously with the document), links are now displayed in a separate window. This improves the performance of browsing considerably, as the users' primary window of interest does not have to wait for links to be resolved. The linking service thus takes on a more advisory nature. This system is implemented by using a proxy that (as seen by the Web browser) acts as a normal proxy but also sends the displayed document to a link server agent that resolves the links relevant to the document. The display of these links is handled by having the AgentDLS browser window request a page from the link server agent with regular intervals.

The Devise Hypermedia group, of which the author is a member, has also made various Web integrations with its Dexter-based hypermedia system [14][15]. The first attempt was DHM/WWW [16]. The architecture consisted of a Java-applet communicating through a CGI-script to a DHM server. When the user requested a document by typing its URL in the applet, the applet would retrieve the document while querying the DHM server for endpoints in the document. The endpoints retrieved was inserted into the Web page as it was being downloaded and displayed in a Web browser window using JavaScript. All links in the Web page were modified so that a click on a link would result in the applet being invoked, allowing it repeat the above described process. The links and endpoints from the DHM server could also be inspected and browsed within the applet. This version had several shortcomings: it was dependent on the user not using bookmarks or entering URLs in the Web browser itself, as such actions would cause the applet to be terminated as its own page would be unloaded. Furthermore it was unable to handle frames (as the loading of a new 'top' frame set would also cause the unloading of the applet) and was limited by the 'sandbox' imposed for security reasons on Java applets[1]. While supported by the DHM server, the DHM/WWW applet could only handle one context (that is one set of hypermedia structures) and had no user concept. A second version, Navette [3], was developed to address some of these issues. Navette was a signed Java applet, allowing the system to use Web pages from arbitrary Web

[1] The Java 'sandbox' security limits a Java applet in various ways. Most crucial to DHM/WWW was the restriction of network contact exclusively to the originating web server, thus making DHM/WWW unable to work with web pages from other web servers.

servers. To speed up communication with the DHM server, TCP/IP and optimistic caching of hypermedia structures (e.g. retrieving a whole context rather than only resolving one link) were used. This version also handled multiple contexts and users. The frame problem remained, and Web pages were still displayed using JavaScript, which made for noticeable degraded performance when browsing with Navette. Simultaneously with Navette, the Webvise client [18], a custom integration with the Microsoft Internet Explorer [28] was being developed. Operating as an application rather an applet removed the limitations put on DHM/WWW and Navette, and using the Microsoft Internet Explorer [28] rather than the Netscape Communicator [31] allowed Webvise to insert links after the browser had displayed the document, thus improving performance considerably. This is done through DOM [11] and the COM-interface available through the Internet Explorer. A second version of Navette has been developed addressing the problems of prior releases using the Arakne framework, which will described below.

Guided tours

Guided tours and trails have been a part of hypermedia from the very beginning [5], when Bush introduced the concept of the trail linking related documents together. Trigg did more recent groundbreaking work in [41]. Several existing systems try to exploit this idea with Web documents.

Walden's Paths [12][37] is a system designed mainly to be used in an educational setting, where a teacher composes trails for students to follow. The teacher uses either the Path Authoring Tool (a Java application) or VIKI [38] combined with a browser as an authoring tool. Trails are stored on a Path Server, which through the use of CGI scripts acts as a proxy while modifying the pages to provide an interface to the path. The interface consists of blocks in the top and the bottom of the page. This block is a graphical representation of (a part of) the path plus additional annotations written by the path author. As all documents go through the Path Servers (links in the documents are modified to achieve this), students can go 'off path' and still return to the path by pressing a button in the interface block. State is communicated by adding arguments into the URL given to the Path Server's CGI-scripts. The Walden's Paths has been extended with regards to collaborative aspects, allowing students to author and share paths of their own. Additionally work has been done to extend upon the linear path by adding conditional branches. The logic to support this is handled by the Path Server, thus still making all functionality accessible from a standard Web browser.

Another Web-based guided tour system is Ariadne [23], which is a Java-based applet. Ariadne operates in an external window to the browser and controls the browser through JavaScript. A guided tour in Ariadne is a directed graph, as opposed to the linear (with branches) path of Walden's Paths. The Ariadne user interface supports both browsing and editing of guided tours. The tours are stored as composites on the Dexter-based DHM [15] server. Leaving the Web pages untouched has several advantages to the Walden's Path approach, as 1) it reduces overhead and complexity as Web documents do not have go through an extra server, and 2) entering URLs or using bookmarks does not pose a problem. On the other hand users are required to use a Java-enabled Web browser rather than any Web browser, though that currently is not a strong requirement. The Ariadne system has recently been adapted to work within the Arakne framework, which will be described in more detail below.

Structuring/Spatial

Spatial hypermedia as described by Marshall & Shipman [24] and as implemented in VIKI [37] is a new kind of hypermedia application, where link structures are no longer explicit but rather implicit based on the spatial relationship between objects. This has become a very powerful tool for organising and structuring, and few hypermedia systems are in more need of organisation and structure than the Web.

Web Squirrel [42] is a URL management system, that uses a spatial metaphor to help users organise their URLs into 'information farms'. The user creates Neighbourhoods onto which URLs are dragged and dropped. The Neighbourhoods and the URLs are arranged spatial as the user wishes, and are analysed by software agents that can create links between URLs according to user's rating of the Web sites and maintain link integrity. The user can create new agents using a scripting language. The information farms are stored locally, but can be distributed to other users of Web Squirrel, exported as HTML, or converted to the Hot Sauce MCF format.

Hot Sauce [21] is a spatial hypermedia plug-in created by Apple. Hot Sauce displays a zoomable 2D representation of a collection of collections and documents. This structure is stored using the XML [47] based Meta Content Format [19], a general format to describe meta content (MCF has thus much wider application than its use in Hot Sauce). Links and collections of links are arranged spatial and the user can zoom in and out, move about, and open collections within the collection. If the user double-clicks on a link, the document is retrieved in a separate window, allowing the user to continue to navigate using Hot Sauce. The Hot Sauce is a media viewer and as such retrieves its MCF file from a Web server.

SUMMARY OF STRATEGIES FOR WEB AUGMENTATION

The tools and systems described above have addressed many problems pertaining to the current Web, and have utilised a lot of different techniques to attempt to solve these problems. The Web augmentation strategies are, using the schema introduced in Table 1, summarised in Table 2. We will below outline some general trends and describe some of the aspects of writing Web augmentation tools that make developing them hard.

Some patterns become apparent. All of the tools reviewed are responsive and need to be aware of the user's actions, be it to record the URL of the current Web page or to perform link and endpoint computations. All need to provide the user with a user interface (though it might be very discreet at most times, as in the 'interfaceless' DLS). Most of the tools need to store data somewhere and most choose to do this on a remote server, thus raising the need to able to communi-

cate over network. The communication is often handled by CGI-scripts, which is problematic, as the tool only gets data when it requests it – the server cannot notify the tool of changes. Many of the tools modify Web pages, and most of the implementations of this functionality would have a hard time interoperating with each other, as they in turn would modify pages and links, quite possibly corrupting each other's data. Most of the Web page modifications are not robust to things such as frames and JavaScript, and many have a problem with forms. The tools relying on modifying link with CGI-scripts rather than using a proxy are fragile to the use of bookmarks or directly entered URLs. The tools that modify Web pages through a proxy are hard to use with other tools that rely on proxies to be informed of the user's actions, unless either of the proxies can it be modified to use the other as a proxy. Furthermore the use of proxies requires the user to modify the Web browser configuration which can be unwieldy if the user does not wish to continually use the tool relying on the proxy. These problems to which there generally are no easy solutions, make it difficult for the developer to create Web augmentation tools.

Tool	Method of Integration	Storage	Web page modification	Collaboration support
ComMentor	Part of browser, CGI-scripts	Remote	Local proxy	Asynchronous
CritLink Mediator	Within browser, JavaScript, forms; CGI-scripts	Remote	CGI-scripts	Asynchronous
Chimera	Web server	Remote	No	Asynchronous
Chimera	Within browser, applet; Web server; cookies, CGI-scripts	Remote	Web server	Asynchronous
Hyper-G	Part of browser/Within browser, forms; web-server;	Remote	Web server	Asynchronous
HyperWave	Within browser, applet, JavaScript, forms; web-server	Remote	Web server	Asynchronous
DLS	Without browser, local; within browser, forms	Remote	No	Asynchronous
DLS	Within browser, forms; proxy	Remote	Proxy	Asynchronous
AgentDLS	Within browser, separate window; proxy	Remote	No	Asynchronous
DHM/WWW	Within browser, applet, JavaScript, CGI-scripts	Remote	Web browser	Shared
Navette	Within browser, applet, JavaScript	Remote	Web browser	Asynchronous
Webvise	Without browser, local	Remote	Web browser	Asynchronous
Walden's Paths	Within browser, JavaScript, forms; CGI-scripts	Remote	CGI-scripts	Shareable
Ariadne	Within browser, applet, JavaScript	Remote	No	Asynchronous
Web Squirrel	Without browser, local	Local	No	Shareable
Hot Sauce	Within browser, plug-in	Remote	No	Shareable

Table 2 - Summary of Web augmentation strategies

TOWARDS A COMMON FRAMEWORK

We are aware of no single "silver bullet", or all encompassing solution, that will solve all the problems described above. However, the similarities between the described Web augmentation tools would suggest that it should be possible to describe and model their functionality in a common framework. This could provide workers in the field with a tool for future conceptual and practical development. Trying to create such a tool, we have come up with the Arakne framework.

The Arakne framework is a conceptual model, which has been implemented as an environment for Web augmentation tools. The implementation is just one implementation of a general framework. The practical issues raised by the summary above will be dealt in the description of the implementation.

The Arakne framework is an object-oriented, component-based, three-layered model aimed at providing Web augmentation tools a unified access to structure servers, proxies, and Web browsers. It is an instantiation of the Common Reference Architecture (CoReArc) for open hypermedia systems, as described by Grønbæk & Wiil [17]. CoReArc divides the architecture of hypermedia systems into three layers: The content layer (displaying and handling documents, displaying structure), the service layer (handling navigation, integration, collaboration etc.), and the structure layer (storage and retrieval of structure).

The Arakne framework is aimed at modelling Web augmentation tools, and the elements contained in the model should now be familiar.

A diagram of the framework can be seen in Figure 1. The framework may support any number of Web augmentation tools. These tools (known as 'navlets') are dependent on four core components of the Arakne framework: the Operations, the Hyperstructure Store, the Browser, and the Proxy. The navlet is the domain specific part of a Web augmentation tool. It provides a user interface as well as special logic to handle the specific domain. This may include deciding which links to display in a Web page based on information retrieved from the Hyperstructure Store component, or interfacing to the Proxy component for analysis of documents. Depending on the situation the computation and analysis may be carried out by the navlet or by another component.

The Operations component models the communication with the structure server layer. This component will thus typically support the same services as the structure server(s). This is where on the wire issues, such as network communication, marshalling, and multiplexing, are handled.

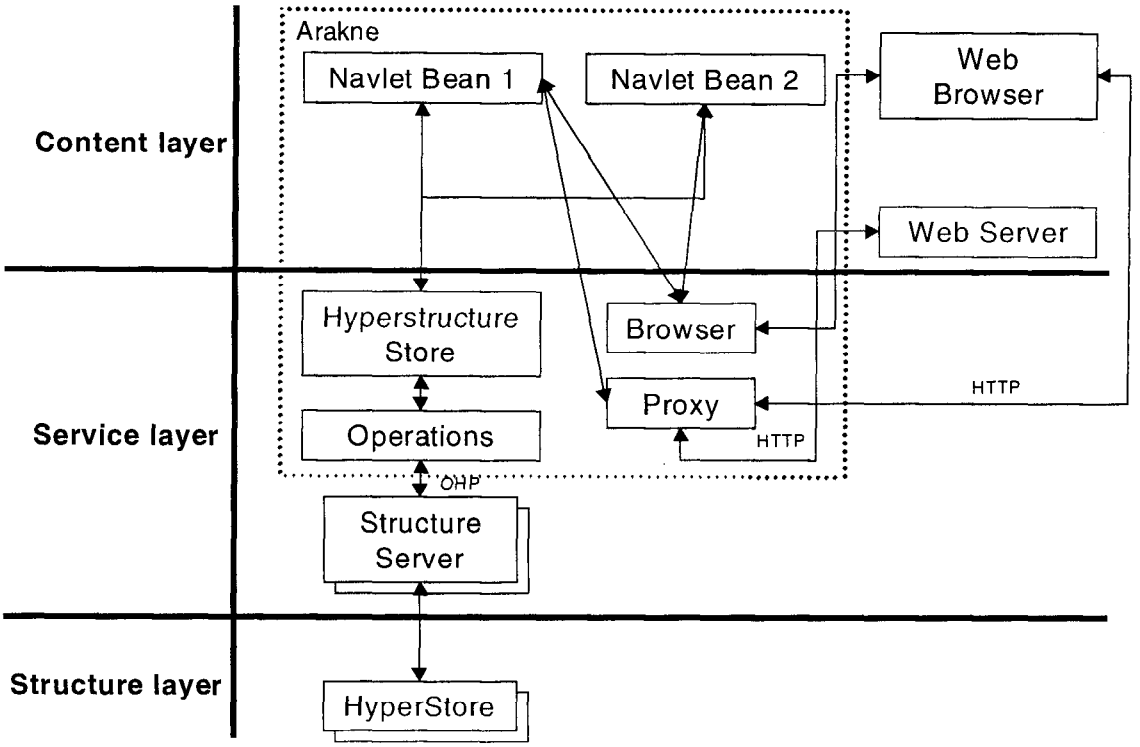

Figure 1 – The Arakne Framework

The Hyperstructure Store is the interface between the navlets and the Operations. The Hyperstructure Store provides convenience functions for the navlets, as well as caching the results of the queries retrieved with Operations. The Hyperstructure Store will also alert navlets to changes in the structures they subscribe to.

The Proxy component models the modification and analysis of Web content. Depending on their domain, navlets may require the Proxy to modify Web pages, and these requests for modifications are collected by the Proxy and used to modify the Web page. Other navlets may require access to the content of a Web page and the Proxy also handles this.

The Browser component models the user's Web browser. Through the Browser navlets can retrieve and modify the state of the Web browser such as which URL is currently displayed; the structure of the current frame set; whether a selection has been made in a frame and if so, what and where.

The situation depicted in Figure 1 is a situation of two navlets, where Navlet Bean 1 is a link creation tool, and thus needs access to the Proxy in order to insert links into Web pages. Navlet Bean 2 is a guided tour tool and does not modify Web pages; and is not connected to the Proxy. Both however need to be able to tell and set the state of the currently displayed documents, as well as retrieving data from the structure server through the Hyperstructure Store.

Mapping Web Augmentation Tools to Arakne

If we review the models of the Web augmentation tools described in this paper, most can be mapped to the abstract Arakne framework. We will in this section argue for the general applicability of the Arakne framework in each of the general Web augmentation categories introduced earlier, and specify the mapping of one representative from each category to the Arakne framework.

Annotation tools are in their functional requirements similar to link creation tools and will dealt with as one. Both need to retrieve hypermedia structures stored on a server, which are handled by the Operations and the Hyperstructure Store components. Many of these tools need to modify Web pages in order to insert links or annotations, which is handled through the Proxy component. Support for collaboration is handled at different levels. The Hyperstructure Store component is able to handle sets of structures as well multiple users. Notifications from the structure servers are handled by the Operations component and forwarded to the Hyperstructure Store component, which notifies navlets, depending on what events they subscribe to.

The ComMentor system, described in [35], has three main elements apart from the structure servers and the Web servers; namely the merge library, the interactive renderer, and the user context control application. The merge library handles the tasks of the Operations, Hyperstructure Store, and Proxy components. The modifications made to the Mosaic browser combined with the code handling it in the user context control application makes up the Browser component.

The AgentDLS [8] architecture consists of a link server agent, that analyses Web pages sent to it by a HTTP proxy and based on this uses link resolvers and link bases to com-

pile a list of relevant links. The role of the proxy is a simple case of the Proxy component with the twist that documents are sent to two parties. The link server agent acts as a combined Operations and Hyperstructure Store component. The analysis performed by link resolvers also would seem to belong to this component, depending on the set-up of the navlet. The Browser component's responsibility – letting the system set or get the state of the Web browser – is handled at two levels: in the Proxy component (what document is displayed now?) and in the AgentDLS window, where users can directly influence, what URL to display in their Web browser.

Guided tour tools require a structure server for storage of the tours as well as the ability to communicate with this server. This can be modelled through the Operations and Hyperstructure Store components. To keep track of where the reader is on the trail (and to put the reader back on track, if need be), it must be possible to set and read the state of a Web browser, which is modelled through the Browser component. Some guided tour tools, e.g. Walden's Paths, modify the Web pages to add comments and interface. While it may not be necessary for interface reasons in the Arakne framework (this could be handled by a navlet), it certainly can be done through the use of the Proxy component.

Could the current implementation of Walden's Path be fitted within the Arakne model? In this case, some of the Arakne components collapse into one. The Path Server acts as the Proxy, Hyperstructure Store, and Operations components, by handling both external structures (guided tours) and Web page modification. The Browser component aspect is handled by the modification of links that keeps the system (i.e. the Path Server) aware of the state of the Web browser. The interface aspects of the navlet are the header and footer of the displayed Web pages that give the user a possibility to interact with the system.

Spatial and structuring Web augmentation tools certainly need structure servers and information about the currently display Web page just as the rest of the above described tools. However, the main focus of the spatial/structure tools described in this paper has been on the user interface and the ability to analyse and process structure and relationships using user written scripts. The user interface is handled by the navlet itself. While the navlet of course would serve as the front-end, the scripting capability fits best within the Hyperstructure Store component, where all structure is stored during run-time and where the Operations component can be directly accessed.

AN IMPLEMENTATION OF THE ARAKNE FRAMEWORK

A system called the Arakne applet has been developed as a part of the Coconut project. The immediate goal of the implementation was to try out the soundness of the Arakne framework by integrating the guided tour tool Ariadne and the link creation tool Navette. The system has undergone some development and has proven robust to the interchange of components.

The system was originally developed as a Java applet running in Netscape Communicator [31]. The components within the dotted line in Figure 1 were a part of this particular implementation. This was an implementation choice; as can be seen from the previous section, the framework itself posits no such requirements on the location of the individual components.

The components in Figure 1 correspond to Java classes in the Arakne applet. The interfaces between components are handled in the MVC (Model View Control) idiom, where the 'view' (e.g. a graph displaying a guided tour) is loosely coupled through events to the actual data, the 'model'. Modifications of data is handled as requests through the 'controller' and if granted notified to the view through the use of events.

The Hyperstructure Store class caches structure retrieved through the Operations class and is the only interface to the structure servers given to the navlet beans. This is done to improve performance (by not retrieving the same information twice) and to ensure data integrity between the Hyperstructure Store and the structure servers. The Hyperstructure Store provides the navlet beans with a rich set of convenience methods, which are translated into queries to the structure servers by the Operations class.

The Browser class encapsulates the needed functionality to communicate with a Web browser, in the original case the Netscape Communicator. The interface is relatively simple and can be adapted to another browser.

The Proxy class is a small proxy running as a thread in the applet. The Netscape Communicator can via JavaScript be set up to use a proxy on the fly, and when the Arakne applet is running, the Proxy class acts as a proxy for the browser. The Proxy class uses whatever proxy the browser was configured to use, and when the Arakne applet is terminated, everything is returned to normal. The Proxy handles requests for Web page modifications and the correct display of frames (thus allowing user to link into frame sets).

All of the classes visible to the navlet beans, the Hyperstructure Store class, the Proxy class, and the Browser class generate events that the navlet beans may subscribe to. The navlet beans are currently restricted to being Java beans and to follow some design guidelines. The current implementation supports only compile time integration of navlet beans giving the user a fixed number of available navlet beans, but future versions of the Arakne applet should be able to load navlet beans on runtime so that the user can retrieve navlet beans from different servers.

The current version of Netscape Communicator[2] has a very limited API for browser integration as well as a faulty Java socket implementation. This lead to the abandonment of the Communicator as the supported Web browser at a time where most components as well as two navlets were finished or nearing completion. The Microsoft Internet Explorer was chosen as the new supported Web browser. This choice has brought some costs, as the Arakne applet is now not only browser-dependent (which was also the case before), but

[2] Version 4.5

now also platform dependent, as Java is not supported in the Microsoft Internet Explorer on anything but the Windows platform.

The migration to the Internet Explorer also served as a (unintended) test of how much code needed to be rewritten to accommodate the change. This code rewrite has been light. The Browser class has been redone, as the Internet Explorer is controlled not through JavaScript, but through a COM interface. Furthermore some unique features such as the ability to operate on selected text in a browser window through the use of right-click menus have been exploited in some changes in user interface, e.g. link creation.

The Internet Explorer does not allow proxy configuration through JavaScript or other means, which has led to some major changes in the Proxy component. The current version relies on the DHMProxy [18] for Web page modification. This is expected to change as the DOM model [11] gives excellent possibilities for Web page analysis and modification, after the Web page has been rendered by the Web browser. This work is expected to be heavily dependent on the experiences from the Webvise application [18].

The current version uses the Devise Hypermedia server as a structure server. This is changing rapidly however, as a new Hyperstructure Store class is being developed to use a new Operations component that utilises the Open Hypermedia Protocol [33]. This has numerous advantages. The OHP is a powerful protocol with a good and general data model. Among the interesting features of OHP is the support for sessions, so that e.g. link following happens simultaneously on several machines as demonstrated at the demo at Hypertext 98. Finally the open hypermedia community supports the OHP and the data model, so that the Arakne applet may communicate with other structure servers such as OHP compliant MicroCosm or HyperDisco servers.

The synchronous collaborative aspects of the Arakne applet have been put on hold until the OHP is fully integrated, though the system as it is supports asynchronous collaboration.

Currently the Arakne applet supports the Ariadne and Navette navlet beans. The user is thus able to create links in documents while creating guided tours, which was not possible before. Each of the navlet beans occupies an internal window in the Arakne applet. The Arakne applet provides the interface for logging on to structure servers and the selection of contexts.

The Navette navlet has recently been extended to support linking to and from segments in video and audio clips through the Mimicry system [4]. Most temporal media plugins do not have APIs well suited for the needs of open hypermedia, which has led to the development of the Mimicry player. The Mimicry player is a Java applet capable of handling most video and audio formats, and it provides a rich API for open hypermedia integration.

Future Plans

Future plans for the Arakne applet include more navlet beans. Currently a spatial navlet bean is under development. The navlets currently supported are not aware of each other, but future versions of Arakne will support inter-navlet communication. The Microsoft Internet Explorer is not expected to be the only Web browser supported by future Arakne applets. The upcoming Mozilla 5.0 holds great promise in this regard. Another area of interest to the developers is the implications of XLink and XPointer.

Experiences of the Development of the Arakne Applet

How does the current Arakne applet compare to the problems raised in the summary of strategies for Web augmentation? Network communication is handled through sockets by the Operations Component, and the client maintains a socket connection, so that the structure servers may contact it. The Arakne applet has so far only supported navlets orthogonal to each other, so there is currently no experiences with regards to two navlets trying to modify the same Web page. However the architecture has only one component, the Proxy component, allowed to modify Web pages, so it should be possible to contain and perhaps avoid most problems. The Arakne applet is currently dependent on a proxy with the usual advantages (all browsing activities are 'captured' by the system) and disadvantages (the user needs to configure the Web browser to use the proxy). The proxy can use other proxies without problems, though the use of two Web page modifying proxies certainly would lead to undefined results. The proxy handles frame sets and inserts links through the use of JavaScript and DOM, so that Web pages are (visibly) modified on arrival rather than in transit, which solves many problem regarding dynamically created documents and frame sets. The Arakne applet is a Java applet with the limitations imposed on Java applets. The security restrictions are handled through digital signing. The Arakne applet is fairly secure from being unloaded by mistake, as it runs in a separate non-resizable browser window with disabled menus and toolbars. A consequence of the integration with the Microsoft Internet Explorer is that a user can not just download and use the Arakne applet. Certain files have to be installed by the user to provide the Arakne applet with right-click menus. This process can however be largely automated.

Some of the solutions found in the Arakne applet are strictly browser and platform dependent. This is very unfortunate, and the only thing that can remedy the situation is a new standard Java API for Web browsers. This API should at least provide functionality similar to that of the API of the Microsoft Internet Explorer, but do so in a browser independent way. Given the current Web browser situation, we think that such an API is unlikely to appear anytime soon, so the next best solution would be a Web browser that provided a platform independent API. Whether or not Mozilla 5.0 will provide such an API remains to be seen.

CONCLUSION

Web augmentation tools will in all probability remain a part of the Web, as researchers and users will continue to explore the boundaries of what hypermedia is and what it can be used to. An understanding of the strategies employed in Web augmentation is needed to make the next generation of Web augmentation tools easier to envision, develop and use.

We have investigated the recurring themes and techniques found in current Web augmentation tools. Based on this, we have developed and described the Arakne framework, and shown that the Arakne framework accommodates existing Web augmentation tasks, and that most tools can be modelled using the framework. A shared framework for these tools could benefit analysis and communication as well as development, and it is hoped that the Arakne framework is a step in the direction of the creation of such a framework.

The interesting and challenging part of creating a Web augmentation tool is not the nitty-gritty of interfacing to a Web browser or writing a proxy. The interesting part, at least in the author's experience, is to create tools that can help users structure their work and their browsing, and by implementing the Arakne applet some of the work needed to provide a full infrastructure for Web augmentation tools have been developed.

The Web has succeeded through (more or less) strict adherence to open standards that have been jointly developed by the interested parties. This has led to the development of standard tools usable for all. The continuation of this trend is crucial to the future development of the Web. The Open Hypermedia System initiative if supported by the hypermedia community can be become a shared standard from which the entire hypermedia community will benefit.

ACKNOWLEDGEMENTS

The author is a member of the Coconut project (http://www.cit.dk/coconut/), a joint research project consisting of Department of Computer Science, Aarhus University and Tele-Danmark Internet. The Danish National Centre for IT-Research (http://www.cit.dk/) supports the Coconut project.

The author wishes to thank Jesper Jühne for valuable discussion of Arakne, René Thomsen for adding to the code, Kaj Grønbæk for valuable advice, and the anonymous reviewers for good suggestions to structure this paper.

REFERENCES

[1] Anderson, K. M., Taylor, R. N., and Whitehead JR., E. J. (1994). Chimera: Hypertext for heterogeneous software environments. In *Proceedings of ECHT 94 Conference*, pp. 97–107, Edinburgh, Scotland.

[2] Anderson, K. M. (1997) Integrating Open Hypermedia Systems with the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 157–166, Southampton, England.

[3] Bouvin, N. O. (1998). Designing Open Hypermedia Applets: Experiences and Prospects. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 281–282, Pittsburgh, USA.

[4] Bouvin, N. O., and Schade, R. (1999). Integrating temporal media with open hypermedia on the WWW. Submitted for publication.

[5] Bush, V. (1945). As we may think. In *The Atlantic Monthly* 176, 1 (July 1945), pp. 101–108.

[6] Carr, L. A., De Roure, D., Hall, W., and Hill, G. (1995). The distributed link service: A tool for publishers, authors and readers. In *Proceedings of the 4th International World Wide Web 95 Conference*, Boston, USA.

[7] Carr, L. A., Hill, G., De Roure, D., Hall, W., and Davis, H. (1996). Open information services. In *Computer Networks and ISDN Systems,* 28, pp 1027–1036, 1996.

[8] Carr, L. A., Hall, W., and Hitchcock, S. (1998). Link services or link agents? . In *Proceedings of the ACM Hypertext 98 Conference*, pp.113–122, Pittsburgh, USA.

[9] Davis, H., Hall, W., Heath, I., Hill, G., and Wilkins, R. (1992). Towards an integrated information environment with open hypermedia systems. In *Proceedings of the ACM Hypertext 92 Conference,* pp. 181–190, Milan, Italy.

[10] CritSuite. http://crit.org/http://crit.org/index.html

[11] Document Object Model. http://www.w3c.org/TR/REC-DOM-Level-1/

[12] Furuta, R., Shipman III, F. M., Marshall, C. C., Brenner, D., and Hsieh, H-W. (1997). Hypertext Paths and the World-Wide Web: Experiences with Walden's Paths. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 167–176, Southampton, England.

[13] Goland, Y., Whitehead, J., Faizi, A., Carter, S., Jensen, D. (1998). Extensions for Distributed Authoring on the World Wide Web – WebDAV. http://www.ics.uci.edu/~ejw/authoring/protocol/draft-ietf-webdav-protocol-08.pdf

[14] Grønbæk, K., and Trigg, R. H. (1994). Design issues for a Dexter-based hypermedia system. In *Communications of the ACM,* 37 (2) February, pp. 40–49, 1994.

[15] Grønbæk, K., and Trigg, R. H. (1996). Toward a Dexter-based model for open hypermedia: Unifying embedded references and link objects. In *Proceedings of the ACM Hypertext 96 Conference*, pp.149–160, Washington DC, USA.

[16] Grønbæk, K., Bouvin, N. O., and Sloth, L. (1997). Designing Dexter-based hypermedia services for the World Wide Web. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 146–156, Southampton, England.

[17] Grønbæk, K., and Will, U. K. (1997). Towards a common reference architecture for open hypermedia. In *JoDI, Journal of Digital Information,* 1(2), 1997. http://jodi.ecs.soton.ac.uk/Articles/v01/i02/Gronbak/

[18] Grønbæk, K., Sloth, L., and Ørbæk, P. (1999). Webvise: Browser and proxy support for open hypermedia structuring mechanisms on the WWW. Submitted for publication.

[19] Guha, R. V., and Bray, T. (Eds.). Meta content framework using XML. http://www.textuality.com/mcf/NOTE-MCF-XML.html

[20] Hall, W., David, H., and Hutchings, G. (1996). *Rethinking Hypermedia: The Microcosm Approach.* Kluwer Academic Publishers, Norwell, USA.

[21] Hot Sauce. http://www.xspace.net/download/index.html

[22] HyperWave. http://www.hyperwave.com/

[23] Jühne, J., Jensen, A. T., and Grønbæk, K. (1998). Ariadne: A Java-based guided tour system for the World Wide Web. In *Proceedings of the 7th International World Wide Web 98 Conference*, Brisbane, Australia.

[24] Marshall, C. C., and Shipman III, F. M. (1995). Spatial hypertext: Designing for change. In *Communications of the ACM,* 38 (8) August, pp. 88–97, 1995.

[25] Marshall, C. C. (1998). Toward an ecology of hypertext annotation. . In *Proceedings of the ACM Hypertext 98 Conference*, pp. 40–49, Pittsburgh, USA.

[26] Maurer, H. (Ed.) (1996). *Hyper-G now, HyperWave: The next generation web solution*, Addison-Wesley, Harlow, 1996.

[27] Meyrowitz, N. (1989). The missing link: Why we're all doing hypertext wrong. In Barrett, E. (ed.). *The society of text: Hypertext, hypermedia and the social construction of information*, pp. 107–114, MIT Press, Cambridge, Massachusetts., USA, 1989.

[28] Microsoft Internet Explorer. http://www.microsoft.com/ie/

[29] Mozilla. http://www.mozilla.org/

[30] NCSA Mosaic. http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/

[31] Netscape Communicator. http://home.netscape.com/download/index.html?cp=djudepart

[32] Nürnberg, P. J., Leggett J. J., Schneider, E., and Schnase, J. L. (1996) Hypermedia operating systems: A new paradigm for computing. In *Proceedings of the ACM Hypertext 96 Conference*, pp.194–202, Washington DC, USA.

[33] The Open Hypermedia Systems Work Group. http://www.ohswg.org/

[34] Rittel, H. and Webber, M. (1973). Dilemmas in general theory of planning. In *Policy Sciences,* Vol. 4, 1973.

[35] Röscheisen, M., Mogensen, C., and Winograd, T. (1995) Beyond browsing: Shared comments, SOAPs, trails, and on-line communities. In *Proceedings of the 3rd International World Wide Web 95 Conference*, Darmstadt, Germany.

[36] Röscheisen, M., Winograd, T., and Paepcke, A. (1995). Content ratings and other third-party value-added information: Defining an enabling platform. In *D-Lib Magazine,* August 1995, http://www.dlib.org/dlib/august95/stanford/08roscheisen.html

[37] Shipman III, F. M., Furuta, R., Brenner, D., Chung, C-C., and Hsieh, H-W. (1998). Using paths in the classroom: Experiences and adaptations. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 267–276, Pittsburgh, USA.

[38] Shipman III, F. M., Furuta, R., and Marshall, C. C. (1997). Generating web-based presentations in spatial hypertext. In *Proceedings of the 1997 International Conference on International User Interfaces*, pp. 71–78, Orlando, USA.

[39] Stanley, T. (1998). Contextures: Focus + context + texture. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 295–296, Pittsburgh, USA.

[40] Trigg, R. H. and Weiser, M. (1986). TEXTNET: A network-based approach to text handling. In *ACM Trans. Office Information. Systems* 4, 1 (Jan 1986), pp 1–23.

[41] Trigg, R. H. (1988). Guided tours and tabletop: tools for communicating in a hypertext environment. In *ACM Trans. Office Information. Systems* 6, 4, pp 398–414, 1988.

[42] Web Squirrel. http://www.eastgate.com/squirrel/Welcome.html

[43] Whitehead Jr., E. J. (1997). An architectural model for application integration in open hypermedia environments. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 1–12, Southampton, England.

[44] Wiil, U. K. and Leggett, J. J. (1996). The HyperDisco approach to open hypermedia systems. In *Proceedings of the ACM Hypertext 96 Conference*, pp.140–148, Washington DC, USA.

[45] Wiil, U. K. and Leggett, J. J. (1997). HyperDisco: collaborative authoring and Internet distribution. In *Proceedings of the ACM Hypertext 97 Conference*, pp. 13–23, Southampton, England.

[46] Wiil, U. K., and Østerbye, K. (1998). Using the Flag taxonomy to study hypermedia system interoperability. In *Proceedings of the ACM Hypertext 98 Conference*, pp. 188–197, Pittsburgh, USA.

[47] XML – Extensible Markup Language. http://www.w3c.org/XML/

[48] Østerbye, K., and Wiil, U. K. (1996). The Flag taxonomy of open hypermedia systems. In *Proceedings of the ACM Hypertext 96 Conference*, pp.129–139, Washington DC, USA.

User Interactions with Everyday Applications as Context for Just-in-time Information Access

Jay Budzik and Kristian J. Hammond
Intelligent Information Laboratory
Northwestern University
1890 Maple Ave.
Evanston, IL 60201 USA
{budzik, hammond}@infolab.nwu.edu

ABSTRACT

Our central claim is that user interactions with everyday productivity applications (e.g., word processors, Web browsers, etc.) provide rich contextual information that can be leveraged to support just-in-time access to task-relevant information. We discuss the requirements for such systems, and develop a general architecture for systems of this type. As evidence for our claim, we present Watson, a system which gathers contextual information in the form of the text of the document the user is manipulating in order to proactively retrieve documents from distributed information repositories. We close by describing the results of several experiments with Watson, which show it consistently provides useful information to its users.

Keywords

Intelligent information access, resource discovery, context, information agent.

1. INTRODUCTION: THE PROBLEM OF CONTEXT

Traditional information retrieval systems [29] have become the cornerstone of information access on the Internet (e.g., [2, 14, 19]) and virtually all other settings in which people access information via the computer. Such systems process requests in the form of query consisting of natural language search terms, and provide the user with a list of links to those documents the system determines are relevant to the query.

From the perspective of the user interface, natural language seems ideal. Users need only express their request in terms of a few search terms. What could be easier? Unfortunately, even if information retrieval systems attempted to understand and represent the concepts being expressed in the documents they index or the requests they process, they are divorced from critical information that is necessary to understand them. Namely, traditional information systems are isolated from the *context* in which a request occurs. Information requests occur for a reason, and that reason grounds the request in contextual information necessary to interpret and process it. Without access to this context, requests become highly ambiguous, resulting in incoherent results, and unsatisfied users.

IUI 2000 New Orleans LA USA

1.1 A Query and Three Scenarios

Consider the request "information about cats" and observe how drastically the utility of various results changes as we manipulate the context of the request in the following scenarios.

Scenario 1: *Veterinary student writing a term paper on animal cancer.* In this case, the most appropriate resources probably have to do with feline cancer, its diagnosis, treatment, etc.

Scenario 2: *Contractor working on a proposal for a new building.* The contractor is most likely referring to Caterpillar Corporation, a major manufacturer of construction equipment, usually shortened to "cat" by people in the construction business.

Scenario 3: *Grade-school student writing a paper about Egypt.* In this case, we would like to see information about cat mummies, laden with pictures and descriptions that are appropriate for a grade school student.

These scenarios illustrate three kinds problems associated with interpreting a request out of context.

Problem 1: *Relevance of active goals.* The active goals of the user contribute significantly to the interpretation of the query and to the criteria for judging a resource relevant to the query.

Problem 2: *Word-sense ambiguity.* The word sense of "cat" is different from the others in scenario 2. The context of the request provides a clear choice of word sense.

Problem 3: *Audience appropriateness.* The audiences in each of the scenarios also constrain the choice of results. Sources appropriate for a veterinarian probably will not be appropriate for a student in grade school.

The above examples uncover several major problems with current information retrieval systems that attempt to process requests out of context. Moreover, a recent study of search engine queries showed that on average, users' queries were 2.21 words long [31]. Needless to say, a two-word query most likely does not contain enough information to discern the active goals of the user, or even the appropriate senses of the words in the query. These problems are not new to researchers in information retrieval. The following section outlines previous work in this area.

2. PREVIOUS EFFORTS IN ESTABLISHING CONTEXT FOR INFORMATION REQUESTS

Previous efforts in establishing context for information access can be classified into four categories: relevance feedback in information retrieval, systems that use user profiles, approaches

based on implicit and explicit techniques for word-sense disambiguation, and knowledge engineering approaches.

2.1 Relevance Feedback

The technique of acquiring relevance feedback [28] to narrow the search space can be seen as a method of discerning context. In systems that support relevance feedback, the user begins with a standard query and then evaluates the results returned (usually by judging a result as relevant or not relevant). The evidence gathered as a result of the user's evaluation is used to modify the original query by adding positive or negative search terms.

In the vector space model of information retrieval [29], queries and documents are represented as vectors in a high dimensional space, where each dimension represents a word or word stem. Given a query Q, a vector in this space, documents D for which the measure $d(Q, D)$ is minimized are first retrieved, where d is some measure of distance (usually the cosine of the angle between the vectors Q and D, or the dot product of the two vectors if the space is normalized). When the user performs a judgment on a document D, the query Q is modified by adding the judged document's terms. E.g., $Q := Q + \alpha D$, where α is a scalar and $|\alpha| < 1$, with a positive value when D is judged relevant, and a negative value when D is judged irrelevant [28].

Few commercial search engines currently support relevance feedback. However, a recent study of Excite [14], one of the only major search engines that did support relevance feedback at the time, showed that users hardly ever took advantage of it [31]. Unfortunately, a different study of the same system [21] showed that users were generally dissatisfied with their results, suggesting that the reason users did not use relevance feedback was not because the information in their initial short query of about two words was sufficient to retrieve relevant results. In addition, this study showed that search sessions did not typically include query refinements (e.g., the addition or deletion of terms), suggesting that users may not be willing to spend the time and effort necessary to use manual or automated refinement techniques. These results argue strongly for methods that automatically perform refinement up-front, instead of requiring explicit user intervention.

2.2 User Profiles

Efforts in building user profiles representing a user's interests can also be seen as a method of gathering contextual information.

User profiles can be collected by gathering terms based on rating documents as in relevance feedback, which was described above. Unlike the relevance feedback techniques, the information gathered in a profile persists across retrieval sessions where it may be automatically added to the user's query [9, 4].

Other systems use machine learning algorithms to induce a classifier for documents based on training examples gathered as a result of a user rating documents [25]. These systems typically learn binary text classifiers that classify documents documents as relevant or irrelevant. Systems that learn Naïve Bayes or Support Vector Machine classifiers are common (see [12] for a good survey).

Systems such as Letizia [24] use implicit feedback in the form of user interactions such as bookmarking a page, to learn a user profile. Letizia uses its user profile to perform lookahead search in the locus of the page the user is currently viewing and recommend links accessible on the current page. Implicit feedback techniques seem particularly promising given that the results of the study discussed above suggest that users are unwilling to provide explicit feedback.

Unfortunately, these kinds of systems have the disadvantage that while they do address the issue of communicating general interests to information systems, they lack access to the user's active goals[1]. In our view, the current goals of the user are more important than long-term interests, especially for providing just-in-time access to relevant information.

2.3 Word-Sense Disambiguation

Some systems attempt to reduce ambiguity by requiring explicit word-sense disambiguation on the part of the user [10], or by using popularity information intrinsic in the structure of hypertext documents [5, 11, 15, 30].

Unfortunately, the disambiguation of word senses only addresses part of the problem of discerning context in an isolated setting. In addition, requiring the user to perform explicit disambiguation may fall prey to the same user interaction problems encountered by explicit relevance feedback techniques. That is, users may be unwilling to choose among multiple senses.

Systems that use reference text to index documents do have an advantage in that they require no intervention on the part of the user, yet they are limited by the fact that they typically use popularity as a metric for ordering results and fall short in taking an analysis of context any deeper. For example, when Google! [15], a system which uses the text of links to index documents, processes the query "gas plasma displays," it will return a long list of documents about blood plasma, because the system has found references which include the word "plasma" most frequently point to documents which discuss blood plasma. Returning to our earlier example, then, just because the most common sense of the word "cat" is a noun representing feline mammals, doesn't mean that this meaning is always what users want.

2.4 Knowledge Engineering Approaches

Knowledge engineering approaches [16, 17, 18, 22] model user behavior in a particular application and explicitly associate queries (or simply a user's actions) in a particular state of the task they are executing with resources that support that task. For example, Argus [22] observes users interacting with a performance support tool and uses a task model in the form of a finite state automaton to detect opportunities to retrieve stories from an organizational memory system.

While the performance of these systems is impressive, they can be difficult to construct and are also usually limited in scope. In essence, they suffer the same problems as designing good hypertext documents: since related documents must be linked explicitly, the designer must have a prior knowledge of the documents to be linked. In settings in which the collection of documents is large and changing (e.g., the Internet) this kind of approach is impractical.

On the other hand, for situations in which user interactions are limited and regular, lexical representations of interface artifacts are unavailable, or in cases where the resources the user is able to access are fixed and limited in number, hand-crafted approaches

[1] Letizia is an exception because it operates on the page the user is currently viewing and attempts to offer assistance in the task of exploring the links on that page by recommending links that are related to the user's interests.

are appropriate. In addition, we see an important opportunity for synergy between knowledge-based approaches and the approach we will describe in the next section that leverage the benefits of both. We will develop this position further in Section 6.

3. INFORMATION MANAGEMENT ASSISTANTS

Our work on Information Management Assistants (IMAs) [6, 8] is strongly motivated by the avenues the above approaches leave unexplored, and by what is known about the behavior of users in information systems.

IMAs observe users interact with everyday applications and attempt to anticipate their information needs using a model of the task at hand. IMAs then automatically fulfill these needs using the text of the document the user is manipulating and a knowledge of how to form queries to traditional information retrieval systems (e.g., Internet search engines, abstract databases, etc.). IMAs embody a *just-in-time* information infrastructure in which information is brought to users as they need it, without requiring explicit requests. IMAs automatically query information systems on behalf of users as well as provide an interface by which the user can pose queries explicitly. Because IMAs are aware of the user's task, they can augment the user's explicit query with terms representative of the context of this task. In this way, IMAs provide a framework for bringing implicit task context to bear on servicing explicit information requests, in an attempt to address the problems associated with processing queries out of context.

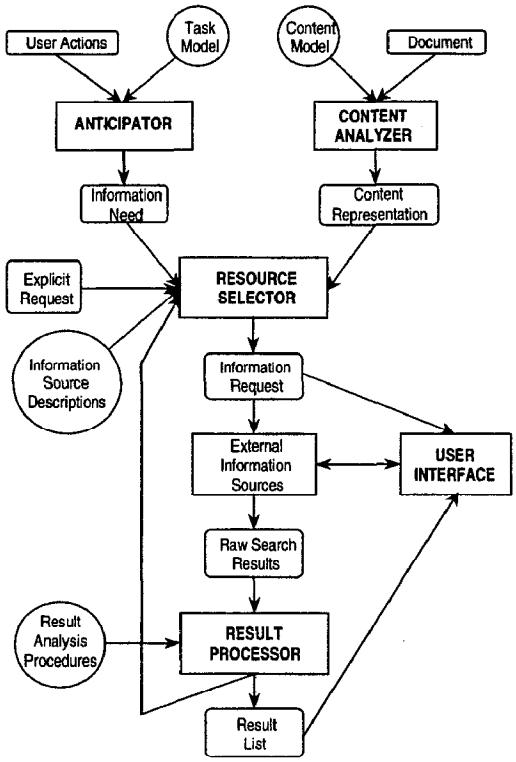

Figure 1: IMA Conceptual Architecture

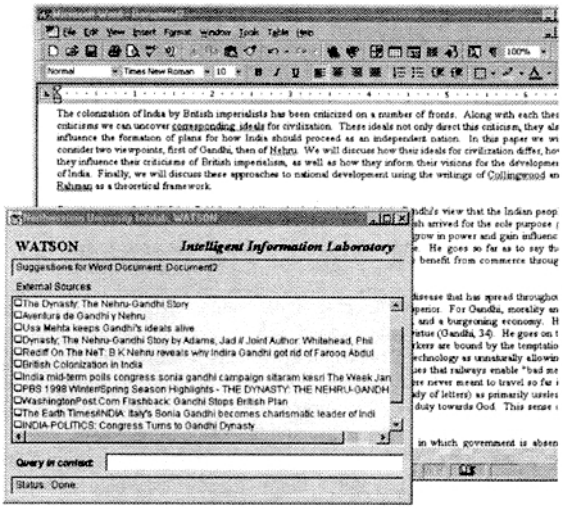

Figure 2: Watson is suggesting documents as a user is writing a paper.

The conceptual architecture for an IMA is displayed graphically in Figure 1. An Information Management Assistant observes users as they interact with everyday applications. The ANTICIPATOR uses an explicit task model to interpret user actions and anticipate a user's possible *information need*. The CONTENT ANALYZER employs a model of the content of a document in a given application in order to produce a *content representation* of the document the user is currently manipulating. This representation is fed to the RESOURCE SELECTOR, which selects information sources on the basis of the perceived information need and the content of the document at hand, using a description of the available information sources. In most cases, this results in an *information request* being sent to external sources. A *result list* is returned in the form of an HTML page, which is interpreted and filtered by the RESULT PROCESSOR using a set of *result analysis procedures*, which may eliminate irrelevant results, or direct the result processor to pose a new request. The resulting list is presented to the user in a separate window.

When the user inputs an explicit query, an IMA uses its knowledge of the user's task context to generate an information request, which is sent appropriate sources, as above. In cases in which it is inappropriate to automatically query information systems and filter the results, information requests can be presented to the user in the form of a button which, when pressed, executes the search and presents the results on-demand.

4. IMPLEMENTATION OF WATSON: AN INFORMATION MANAGEMENT ASSISTANT

A preliminary version of this architecture is realized in Watson, the first IMA we have built [6, 8]. Watson has several *application adapters*, which are used to gain access to an application's internal representation of a document. The adapters produce a document representation, which is sent to the Watson application when deemed necessary. Documents are represented as sequences of words in one of four styles: normal, emphasized, de-emphasized or list item.

Next, using a knowledge of the kind of information need anticipated, Watson transforms the original document representation into a query, and selects appropriate sources. This query takes the form of an internal query representation, which is then sent to selected *information adapters*. Each information adapter translates the query into the source-specific query language, and executes a search. Information adapters are also responsible for collecting the results, which are gathered and clustered using several heuristic result similarity metrics, effectively eliminating redundant results (due to mirrors, multiple equivalent DNS host names, etc.). Figure 2 demonstrates the interface associated with the execution of these two processes in sequence.

In parallel, Watson attempts to detect conceptually atomic, lexically regular structures in the document. Once such objects are detected, Watson presents the user with a common action for the item in the form of a button they can press. For example, if a page contains an address, Watson will present a button allowing the user to access a map for the address.

The following sections describe heuristics for the analysis of the gross structure of documents (e.g., word emphasis, or list membership) to the end of automatically constructing queries to information sources, as well as fine-level analysis aimed at detecting conceptually atomic, lexically regular structures. We also describe an algorithm for filtering search results aimed at eliminating redundant results and detecting broken links.

4.1 Term Weighting for Query Construction

In order to retrieve related documents as the user is writing or browsing, Watson must construct a query based on the content of the document at hand that will eventually be sent to external information sources in real time. Text retrieval systems typically require queries in the form of a sequence of search terms or keywords. The following heuristics were useful in constructing an algorithm that extracts search terms from a document to be included in such a query.

Heuristic 1: Remove stop words. Words included in a stop list are not good search terms. They will be automatically removed by the information systems themselves.

Heuristic 2: Value frequently-used words. Words used frequently are representative of the document's content.

Heuristic 3: Value emphasized words. Emphasized words are more representative of the document's content than other words. Emphasized words are used in titles, section headings, etc., and also draw more attention to the document's reader.

Heuristic 4: Value words that appear at the beginning of a document more than words that occur at the end. Because reading is a linear process, words that occur earlier on tend to be descriptive of the rest of the document.

Heuristic 5: Punish words that appear to be intentionally de-emphasized. Words in small fonts (de-emphasized words) are exempt from Heuristic 4.

Heuristic 6: Ignore the ordering of words in lists. Words found in lists are a special case, because they are intentionally ordered, and are therefore exempt from Heuristic 4.

Heuristic 7: Ignore words that occur in sections of the document that are not indicative of content. Words that occur in the navigation bar of Web page are only marginally useful, and tend to get in the way of text analysis.

```
for each word w collected
    for each (position p, style s) in pos(w)
        weight := 1 + (numTerms^2 δ)/p^2
        if (weight > maxCount) or
            (s is the list item style) or
            (s is the de-emphasized style) then
                weight := 1
        else if (s is the emphasized style) then
                weight := 2 weight
        weight(w) := weight(w) + weight
```

Figure 4: Term Weighting Algorithm

The above heuristics were used to construct the following document representation and term weighting algorithm.

Documents are represented in terms of an ordered list of words in one of three styles. Words can either be normal, emphasized, de-emphasized, or list items. Words are classified into these groups (or omitted, per heuristic 7) by detecting the appropriate structures in HTML documents (for Internet Explorer), or by using the word properties provided by Microsoft Word. Each of Watson's application adapters sends a typed message containing a sequence of words of that type, represented as a string, to the Watson application. Watson then tokenizes the string using a basic lexical analyzer, removes stop words, and sends each term through the following weighting algorithm. Simultaneously, Watson sends tokens to an array of conceptual unit detectors described later in this paper.

The pseudocode displayed in Figure 3 describes the term weighting algorithm. A first pass through the terms has eliminated the stop words, and computed general statistics. At this point, *maxCount* is defined as the maximum number of times any one word has appeared in the document, *numTerms*, is defined as the total number of terms that were not stop words in the document, δ is constant factor, usually defined as 0.2, and pos(w) contains a list of position-style pairs for a given word w.

Intuitively, the preliminary weight of a term varies inversely with the square of its position on a page. This metric improves as the document length increases, prompting the addition of the numerator (which is proportional to a fraction of the square of the total number of terms) to reflect this fact. The term's final weight is the sum of the preliminary weights that are less than *maxCount*, unless the term is a list item or de-emphasized. If a term occuring in a given position is a list item or de-emphasized, its preliminary weight is 1. If a term is emphasized, its preliminary weight is double the original preliminary weight.

The resulting term-weight pairs are sorted, and the top 20 terms are reordered in the order in which they originally occurred in the document. This ordered list is then used to form a query that is sent to selected information sources.

4.2 Result Clustering

Because the results returned from traditional information systems often contain copies of the same page or similar pages from the same server, an IMA must filter the results so as not to *add* to a user's feeling of information overload. If these similarities are not accounted for, some of the more useful pages returned by the traditional information systems may be missed. Moreover, we constantly face the risk of annoying the user instead of helping her. As a result, we actively attempt to reduce the amount of

irrelevant information presented. To this end, Watson collects search engine results and clusters similar pages, displaying a single representative from each cluster for the user to browse.

For the task of clustering redundant results, Watson uses two pieces of information that sources return for each document: the document's title, and its URL. It employs the following heuristic similarity metrics for each of these pieces of information:

Heuristic 1: Title similarity. Two titles are similar if they have a large percentage of words in common. The certainty of similarity increases as a function of the square of the length of the title in words.

Heuristic 2: URL similarity. Two URLs are similar if they have the same internal directory structure. The certainty of similarity increases proportionally as a function of the square of the length of the URL in directory units.

The combination of these similarity metrics is generally sufficient for approximating the uniqueness of the documents returned. Note that we do not perform a clustering based on the full text of the document, as this would be far too bandwidth intensive given the goal of producing results in real-time.

The clustering algorithm we use is incremental. That is, when a new response arrives from the network, it is immediately processed, and the resulting list of suggestions is updated and presented. The aim is to minimize the delay between receiving a response and updating the user interface. In general, the idea is to allow the user to access updated information as soon as it is available. As a result, the user is able to access a site in the middle of the clustering process, even if the system is waiting for further results to be returned from the information sources. As the suggestion list is being collected and incrementally computed, more detailed and expensive processing of the list (such as URL validity checking) is performed in the background, as a separate thread. We call this approach *Present First, Process Later* in the tradition of Riesbeck's "Shoot first, ask questions later" paradigm of anytime reasoning [27].

4.3 Detecting Opportunities to Provide Special-Purpose Information

Watson must be able to reason about the contents of a document well enough to provide helpful suggestions. The previous section described an algorithm for computing a query that will be sent to online information sources, based on the text of a document. While this is helpful, it is only one of several things an Internet browsing or document composition assistant might do. In particular, we would like our assistant to be able to recognize opportunities to provide assistance by completing queries to special-purpose information repositories. To this end, Watson has a facility for detecting lexically regular, conceptually atomic items (such as addresses or company names) and providing the user with an interface to useful special-purpose information resulting from a query to specific kinds of online information sources.

In order to detect conceptual units for special purpose search, Watson runs an array of simple detectors in parallel. Each detector is a finite state automaton accepting a sequence of tokens representing a conceptual unit. When a conceptual unit is detected, Watson presents the user with a common action for the item in the form of a button they can press. For example, when Watson detects an address, it presents a button which, when pressed, will display a web page with a map for that address using an automated map generation service.

Watson also detects opportunities for performing special-purpose search in the context of document composition. For example, when a user inserts a caption with no image to fill it in their Microsoft Word document, Watson uses the words in the caption to form a query to an image search engine. Users can then drag and drop the images presented directly into their document.

4.4 Processing Explicit Queries in Context

Watson processes explicit queries in the context of the document the user is currently manipulating. Watson's list of collected results makes its representation of the task context visible to the user. This kind of visibility is usually absent from most systems based on user profiles, yet it is extremely valuable because it allows users to form coherent expectations about the system's performance. It is important to note that this representation not only provides the user with valuable information about otherwise hidden system states instrumental in forming expectations about their next interaction with the software, but it also represents information that useful in and of itself.

When a user submits a query to Watson, it combines the new query terms with the previously constructed contextual query by concatenating them to form a single query. In this way, Watson brings the previously gathered context information to bear directly on the process of servicing a user's explicit query. The addition of the terms Watson has gathered serves to make the user's active goals explicit to the information service, reduce word-sense ambiguity, as well as ensure audience-appropriateness.

For example, if a user is viewing a construction equipment vendor's page and enters the query "toy", Watson will return a list of pages for companies selling model construction equipment.

In addition to interfacing Watson with traditional information systems, we have also experimented with Watson as a context-bearing interface to Q&A [7]. Q&A is a system for capturing, organizing, and accessing a memory of questions and answers. When a user poses a question to the system, Q&A attempts to answer it by retrieving similar questions. If the question is unanswered, it is forwarded to an appropriate expert, who then answers the question. The user is then notified, and the resulting question and answer are captured and indexed in the system. Watson adds value to a system like Q&A by grounding incoming questions in the context of the document they are writing and the user's browsing history. Given this contextual information, the expert responsible for answering an incoming question is more likely able to grasp the meaning of a possibly ambiguous query and, likewise, is better able to answer the question.

5. EVALUATION OF WATSON

An evaluation was performed in order to determine whether or not the sources returned by Watson were *useful* in the context of a particular task. Because Watson is intended to work alongside the user as she is completing a task, evaluating the utility of the information provided is more appropriate than the relevance-based judgments that are typical of most other evaluations of information retrieval systems.

For this evaluation, we asked 10 researchers in the Computer Science department to submit an electronic version of the last paper they wrote. Each paper was loaded into Microsoft Word while Watson was running. The results Watson returned were then sent back to the authors of the paper. Subjects were asked to judge whether or not the references would have been useful to them when they were preparing the paper. 8 out of 10 subjects

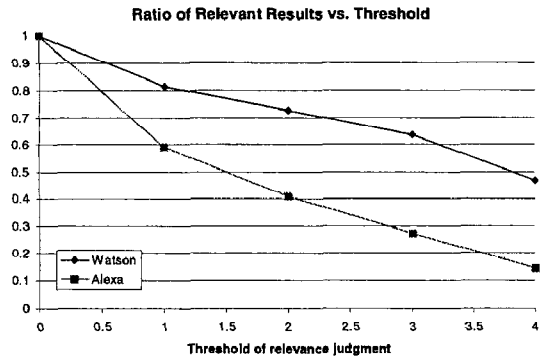

Figure 5: Relevance ratings for suggestions provided by each system were given on a 5-point scale. Shown above is the ratio of relevant results as a function of the threshold above which a document is considered relevant.

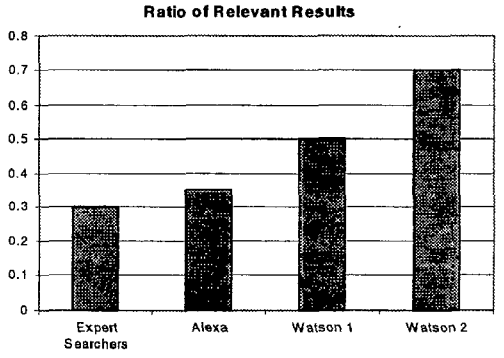

Figure 6: Comparison of Expert searchers, Alexa and two versions of Watson. Data for expert searchers and Watson 1 was gathered using a binary scale, while data for Alexa and Watson 2 was gathered using a 5-point scale. In the latter case, references given a rating >2 were counted as relevant.

indicated that at least one of the references returned would have been useful to them. In addition, 4 of the subjects indicated the references Watson provided were completely novel to them, and would be cited or used in their future work. At the same time, some of the pages returned were *mere-appearance matches:* they were lexically similar, but generally off-topic. Future work on improving query construction as well as result filtering is aimed at addressing the failure modes uncovered in this and other studies.

A comparison study was also performed in order to evaluate the recommendation portion of Watson. For this study, we collected a list of pages from other researchers at Northwestern. We then asked users to choose a page from the list, look at it in a Web browser and then use Alta Vista to find similar pages. The users then judged the top 10 pages returned as relevant or irrelevant to their search task. Next, the users were asked to judge the sites Watson returned from the same page in the same way. In this experiment, Watson used Alta Vista as well. For our initial group of subjects, we drew from local computer science graduate students. All of the volunteers considered themselves expert-level searchers. This was evident in their query behavior, as most of them used long queries (≥4 words), laden with advanced features. We gathered 19 samples from a pool of 6 users. Using Alta Vista, our group of expert searchers was able to pose queries that returned, on average, 3 relevant documents out of 10. Watson was able to do considerably better at the same task, returning, on average, 5 relevant documents out of 10. In the samples gathered, Watson was able to do as well or better than an expert user 15 out of 19 times.

A further comparison was performed of Watson and Alexa [1] in order to determine whether or not Watson provided a significant improvement over the recommendations currently available in commercial systems. Alexa is a system that recommends Web pages given a URL as input, and is freely available to Internet users. It is unclear how Alexa works, because a detailed description of the system is not available. It is important to note, however, that we can only compare Alexa *as a system* with Watson using other information sources *as a system*. We cannot make claims, here, about the effectiveness of particular *algorithms* because we do not have control over the contents of the systems' databases.

This said, for this experiment, we gathered a collection of URLs from a volunteer's bookmarks. We then had a volunteer evaluate the relevance of the recommendations returned by Alexa and by Watson on a 5-point scale. The version of Watson used in this experiment was using both Alta Vista and the Google [15] search engine. The volunteer judged recommendations from 15 URLs, which resulted in performing 316 ratings altogether. For the following summary statistics, ratings of 3 and above were considered relevant. Using this criterion for relevance, Alexa returned on average 4 relevant documents out of 10. Watson, was able to do significantly better, returning on average more than 7 relevant documents out of 10. In addition, Watson was able to return 118 relevant documents, while Alexa only returned 54. The data from this study are summarized in Figure 5. As the chart indicates, this experiment showed that as the definition of relevance increases in strictness, the gap between Watson and Alexa's performance widens.

A summary of both of the comparison studies is displayed in Figure 6. This graph should be taken with a grain of salt, as the data displayed are gathered from two different experiments, using two different survey methodologies. However the data suggest that Watson significantly outperforms both Alexa and expert users in providing relevant recommendations for the URLs in the experiments. The improvement in the second version of Watson indicates that Watson's performance is related to the information source it queries, and that the inclusion of Google seems to improve the recommendations it gives.

While this initial evaluation is promising, an expanded evaluation of this and the other features is needed. Moreover, some of the pages used in the evaluation contained structures that Watson currently does not handle very well. For example, some of the pages in the evaluation pool were frame sets—collections of pages that should be treated as a unit. Watson currently treats each frame as a separate page. Despite this limitation, Watson performed quite well.

6. CLOSELY RELATED WORK

Software that recommends Web pages and learns user preferences has been an intense focus of must recent research. Closely related work that has not been discussed previously in the present paper includes Metasearcher [3] a system which uses a collection of

browser caches gathered from users working in collaboration on a common research task to form queries that are sent to search engines, the results of which are analyzed using LSI [12]; and The Remembrance Agent [26], a system that suggests similar documents as a user composes a new document by performing IR search against a local corpus of previously written text.

Watson is different from Metasearcher in that it works online and focuses on providing just-in-time access to relevant information, whereas Metasearcher performs compute- and network-intensive analyses that are inapplicable in this context. Watson is different from the Remembrance Agent in that it uses external information repositories and in its techniques for selecting representative terms. Finally, Watson goes a step beyond either by providing an interface for communicating explicit requests to the system that are processed in the context of current activity, as well as by providing an architecture for recognizing opportunities to provide special-purpose information. Both of these functionalities represent a fundamental belief underlying our approach: that while it is a good start, similar information is not necessarily useful or even relevant information. We will return to this point in the next section.

7. DISCUSSION AND FUTURE WORK

Watson's representation of a task context is a collection of words associated with the user's current document. Future work will include augmenting this representation to include task and document models, as well as a user profile that could allow Watson to process ambiguous requests in the absence of a discernable task.

In addition, we are working on incorporating semantic knowledge of particular tasks with the aim of improving query construction and source selection. Underlying this effort is the view that similar information is not always the most relevant information. For example, in collaboration with Larry Birnbaum and Marko Krema, the Watson framework was used to build a prototype system called Point-Counterpoint, which assists users in supporting their point of view while they are developing a written argument. The system is based on the idea that when formulating an argument in support of a particular point, other documents which represent arguments both for *and against* that point are useful references. Point-Counterpoint uses knowledge of opposing experts in particular domains to recognize opportunities to retrieve examples of contrary points of view. For example, when a user cites Marx's idea of an ideal economic state, Point-Counterpoint will retrieve two sets of articles: one set representing Marx's point of view, and another set representing Adam Smith's opinion. The queries Point-Counterpoint forms are composed of two distinct sets of terms—*expert terms* and *issue terms*. These queries are formed by modifying the result of the Watson query generation algorithm described above by substituting the name of an expert with his opposite while retaining the terms that represent the general topic of the argument.

This view exemplifies the notion that queries should be treated as first class representational objects that can be modified and transformed by knowledge-based systems in service of a user's information need. Vector space representations of documents are particularly good for computing document-to-document similarity in large collections. We believe coupling such representations with semantic knowledge of a particular task or theme is an exiting avenue of future investigation. The Information Management Assistant architecture provides for such a coupling.

8. CONCLUSION

In summary, we have outlined several problematic issues associated with contemporary information access paradigms, the first and foremost of which is that information systems are divorced from contextual information necessary to coherently process a short request. In response to this, and in light of previous attempts, we presented an architecture for a class of systems we call *Information Management Assistants*. These systems observe user interactions with everyday applications, anticipate information needs, and automatically fulfill them using Internet information sources. An IMA's query is grounded in the context of the user's tasks. IMAs effectively turn everyday applications into intelligent, context-bearing interfaces to conventional information retrieval systems. We presented an overview of our work on Watson, a prototype of this kind of system. We then described an evaluation that underscored the effectiveness of our approach. Finally, we closed with directions for future research.

Information Management Assistants embody a vision of a future in which users hardly ever form a query to request information. When an information need arises, a system like Watson has already anticipated it and provided relevant information to the user before she is even able to ask for it. Failing this, a user could explicitly express information needs to the system, which would service her request within the context of the current task.

By providing a framework for leveraging the context of a user's task, we believe IMAs provide a compelling new framework for research in intelligent information systems.

9. ACKNOWLEDGEMENTS

This work benefited from the efforts of Andrei Scheinkman, Cameron Marlow, and Justin Ramos who helped in the implementation of Watson. In addition, we thank Larry Birnbaum, Shannon Bradshaw, Louis Gomez, Jim Jansen, Chris Riesbeck, Amanda Spink, and anonymous reviewers for candid discussions and helpful suggestions regarding this work.

10. REFERENCES

[1] Alexa. http://www.alexa.com/

[2] Alta Vista. http://www.altavista.com/

[3] Badue, C., Vaz, W., and Albuquerque, E. Using an Automatic Retrieval System in the Web to Assist Co-operative Learning. In *Proc. 1998 World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[4] Billsus, D. and Pazzani, M. A Personal News Agent that Talks, Learns and Explains. In *Proceedings of the Third International Conference on Autonomous Agents.* ACM Press, 1998.

[5] Bradshaw, S., and Hammond, K. J. Mining Citation Text as Indices for Documents. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[6] Budzik, J, Hammond, K., Marlow, C., and Scheinkman, A. Anticipating Information Needs: Everyday Applications as Interfaces to Internet Information Sources. In *Proceedings of WebNet-98, Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[7] Budzik, J., and Hammond K. Q&A: A system for the Capture, Organization and Reuse of Expertise. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[8] Budzik, J., and Hammond, K. Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[9] Chen, L., and Sycara, K. WebMate: A Personal Agent for Browsing and Searching. In *Proceedings of Agents-98, the Second International Conference on Autonomous Agents.* ACM Press, 1998.

[10] Cheng, I., and Wilensky, R. *An Experiment in Enhancing Information Access by Natural Language Processing.* Technical Report CSD-97-963, Computer Science Division, University of California, Berkeley, 1997.

[11] Dean, J., and Henzinger, M. R. Finding related pages in the World Wide Web. In *Proceedings of WWW-8, the Eighth International World Wide Web Conference.* Fortec Seminars, 1999.

[12] Dumais, S. T., G. W. Furnas, T. K. Landauer, S. Derrwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of CHI-98, Conference on Human Factors in Computing Systems.* ACM Press, 1988.

[13] Dumais, S. T., Platt J., Heckerman, D., and Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Classification. In *Proceedings of ACM-CIKM98.* ACM Press, 1998.

[14] Excite. http://www.excite.com/

[15] Google. http://www.google.com/

[16] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* AAAI Press, 1998.

[17] T. Lau and E. Horvitz, Patterns of Search: Analyzing and Modeling Web Query Refinement. In *Proceedings of the Seventh International Conference on User Modeling.* ACM Press, 1998.

[18] D. Heckerman and E. Horvitz. Inferring Informational Goals from Free-Text Queries. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* AAAI Press, 1998.

[19] Lycos. http://www.lycos.com/

[20] Jaczynski, M., and Trousse, B. WWW Assisted Browsing by Reusing Past Navigations of a Group of Users. In *Proceedings of EWCBR-98, European Workshop on Case-Based Reasoning.* Springer, 1998.

[21] Jansen, B., Spink, A., and Bateman, J. Searchers, the Subjects they Search, and Sufficiency: A Study of a Large Sample of EXCITE Searches. In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[22] Johnson, C., Birnbaum, L., Bareiss, R., and Hinrichs, T. Integrating Organizational Memory and Performance Support. In *Procedings of IUI-99.* ACM Press, 1999.

[23] Joachims, T., Freitag, D., and Mitchell, T. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* AAAI Press, 1996.

[24] Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. In *Proceedings of IJCAI-95, International Joint Conference on Artificial Intelligence.* July 1995.

[25] Pazzani, M., Muramatsu J., & Billsus, D. Syskill & Webert: Identifying interesting web sites. *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* AAAI Press, 1996.

[26] Rhodes, B., and Starner, T. A continuously running automated information retrieval system. In *Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology.* April 1996.

[27] Riesbeck, C. What Next? The Future of Case-based reasoning in Post-Modern AI. In Leake, ed., *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* New York, NY: AAAI/MIT Press, 1996.

[28] Salton, G., and Buckley, C. Improving Retrieval Performance by Relevance Feedback. In Spark Jones and Willet, (Eds.) *Readings in Information Retrieval.* San Francisco, CA: Morgan Kauffman, 1990.

[29] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Communications of the ACM* 18(11):613-620, 1971.

[30] Spertus, E. Parasite: Mining Structural Information on the Web. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[31] Spink, A., Bateman, J., and Jansen, B. Users' Searching Behavior on the EXCITE Web Search Engine. In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

Selecting Task-Relevant Sources for Just-in-Time Retrieval*

David B. Leake and Ryan Scherle
Computer Science Department
Indiana University
150 S. Woodlawn Avenue
Bloomington, IN 47405
{leake,rscherle}@cs.indiana.edu

Jay Budzik and Kristian Hammond
The Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue
Evanston, IL 60201, U.S.A.
{budzik,hammond}@ils.nwu.edu

Abstract

"Just-in-time" information systems monitor their users' tasks, anticipate task-based information needs, and proactively provide their users with relevant information. The effectiveness of such systems depends both on their capability to track user tasks and on their ability to retrieve information that satisfies task-based needs. The Watson system (Budzik *et al.* 1998; Budzik & Hammond 1999) provides a framework for monitoring user tasks and identifying relevant content areas, and uses this information to generate focused queries for general-purpose search engines and for specialized search engines integrated into the system. The proliferation of specialized search engines and information repositories on the Web provides a rich source of additional information pre-focused for a wide range information needs, potentially enabling just-in-time systems to exploit that focus by querying the most relevant sources. However, putting this into practice depends on having general scalable methods for selecting the best sources to satisfy the user's needs. This paper describes early research on augmenting Watson with a general-purpose capability for automatic information source selection. It presents a source selection method that has been integrated into Watson and discusses general issues and research directions for task-relevant source selection.

Introduction

As the volume of available information grows, the burden of information access grows as well. "Just-in-time" (JIT) information systems address this problem by shielding the user from the information access task. Instead of requiring a user to recognize the need for information and initiate queries to satisfy it, these systems observe the user's actions in a task context, anticipate the user's information needs, gather the needed information and present it to the user before the user requests it. Such systems require methods for (1) determining the type of information the user requires, and (2) focusing retrieval on information that satisfies the user's needs.

There are now large numbers of focused information sources on the web, providing a rich range of specialized information aimed at satisfying particular information needs.[1] Finding the right sources itself requires expertise, limiting the usefulness of these systems for non-expert users. However, if their information can be provided automatically, this drawback is nullified. This paper describes ongoing research on enabling just-in-time information systems to automatically select information sources that are appropriate to the user's needs.

We are investigating this problem with SourceSelect, a source selection system integrated with the Watson system (Budzik *et al.* 1998; Budzik & Hammond 1999). Watson automatically fulfills users' information needs by monitoring their interactions with everyday applications, anticipating their information needs, and querying Internet information sources for that information. The initial version of Watson focuses on identifying task-relevant content areas and automatically generating content-relevant queries for general-purpose search engines and a small set of specific search engines associated to particular query types by hand-made strategies. SourceSelect provides an initial approach to adding a general-purpose capability for identifying and accessing content-relevant search engines. Given a query from Watson, the system does a two-step retrieval, first using vector-space retrieval methods to associate queries to relevant sources, and then using automatically-generated queries to guide search within those sources. In the combined system, Watson monitors user activities, identifies relevant content areas, and provides SourceSelect with context information. The SourceSelect system determines appropriate information sources, formulates queries to those

*David Leake gratefully acknowledges the support of the InfoLab and Computer Science Department of Northwestern University during his sabbatical leave. His research is supported in part by NASA under award No NCC 2-1035. Ryan Scherle's research is supported in part by the Department of Education under award P200A80301-98. The research of Kristian Hammond and Jay Budzik is funded by a grant from the National Science Foundation, McKinsey and Company, and gifts from Microsoft Research.

[1] For a sampling of some of these, see The Scout Report (http://wwwscout.cs.wisc.edu/scout/report).

sources, sends off those queries, and collates their results for Watson to pass them on to the user. No user intervention is required to target candidate sources.

The paper begins by sketching the Watson framework and discussing the value of specialized information source selection. It next describes the system and the source selection methods it implements. It then discusses central issues for intelligent source selection and how the approach relates to other current approaches.

Just-in-Time Information Access: The Watson Framework

The Intelligent Information Laboratory (InfoLab) at Northwestern University is developing a class of systems called Information Management Assistants (IMAs). These systems observe users as they go about completing tasks in everyday software applications and uses its observations to anticipate the user's information needs. They then automatically fulfill these needs by querying traditional information sources such as Internet search engines, filtering the results and presenting them to the user. IMAs embody a just-in-time information infrastructure in which information is brought to users as they need it, without requiring explicit requests. Essentially, they allow these applications to serve as interfaces for information systems, paving the way for removing the notion of query from information systems altogether.

The first IMA developed at the InfoLab is Watson, an IMA that observes user interaction with applications such as Netscape Navigator, Microsoft Internet Explorer, and Microsoft Word. From its observations and a basic knowledge of *information scripts*—standard information-seeking behaviors in routine situations—Watson anticipates a user's information needs. It then attempts to automatically fulfill them using common Internet information resources.

The conceptual architecture for IMAs has four components (Budzik *et al.* 1998):

- The ANTICIPATOR uses an explicit task model to interpret user actions and anticipate a user's information needs.
- The CONTENT ANALYZER employs a model of the content of a document in a given application in order to produce a content representation of the document the user is currently manipulating.
- The RESOURCE SELECTOR receives the representation produced by the CONTENT ANALYZER and selects information sources on the basis of the perceived information need and the content of the document at hand, using a description of the available information sources. In most cases, this results in an information request being sent to external sources. A result list is returned in the form of an HTML page.
- The RESULT PROCESSOR interprets and filters the result list. Results are gathered and clustered using several heuristic result similarity metrics, effectively eliminating redundant results (due to mirrors, multiple equivalent DNS host names, etc.). The resulting list is presented to the user in a separate window.

The above mechanism allows Watson to suggest related information to a user as she writes or browses the Web. Watson observes user interaction with Microsoft Word and Internet Explorer, and uses information sources ranging from general-purpose information repositories such as newspaper archives or AltaVista, to special-purpose information sources such as image search engines and automatic map generators.

When a user navigates to a new Web page, Watson suggests pages related to the topic of the page at hand. Similarly, as a user composes a document in Microsoft Word, Watson suggests Web pages on the topic of the document she is composing. This is illustrated in Figure 1.

Motivations for Automatic Source Selection

A well-known problem in generating Internet searches is that queries usually return a wide range of information that may not be relevant to user tasks. For the query "home sales," for example, the first page of results for a recent query to AltaVista contained pointers to information on real estate, realtors and mortgages. This is useful information if the user is interested in the mechanics of selling a home. However, if the user is an economist interested in economic indicators, these references are of little use.

If the context for the "home sales query" is known to be that the user is working on a document on economics, it is possible to anticipate the type of result that will be useful. One way to do this is to add additional search terms. This can be useful, but it is sometimes difficult even for an expert to select the right query terms for the desired subset of information to be retrieved.

Sending queries to specialized search engines makes it possible to delineate context in advance of the query itself. A search engine such as CNN financial, for example, provides a focus towards financial news, and sending the "home sales" query there yields the information an economist might want: information on changes in aggregate sales trends.

The number of specialized search engines and repositories is large and rapidly increasing, providing the opportunity to select task-relevant sources to improve search results—if the right sources can be found. Unfortunately, finding the right sources can itself require considerable expertise. However, if a system such as Watson could automatically provide information from the right sources, the usefulness of its results could po-

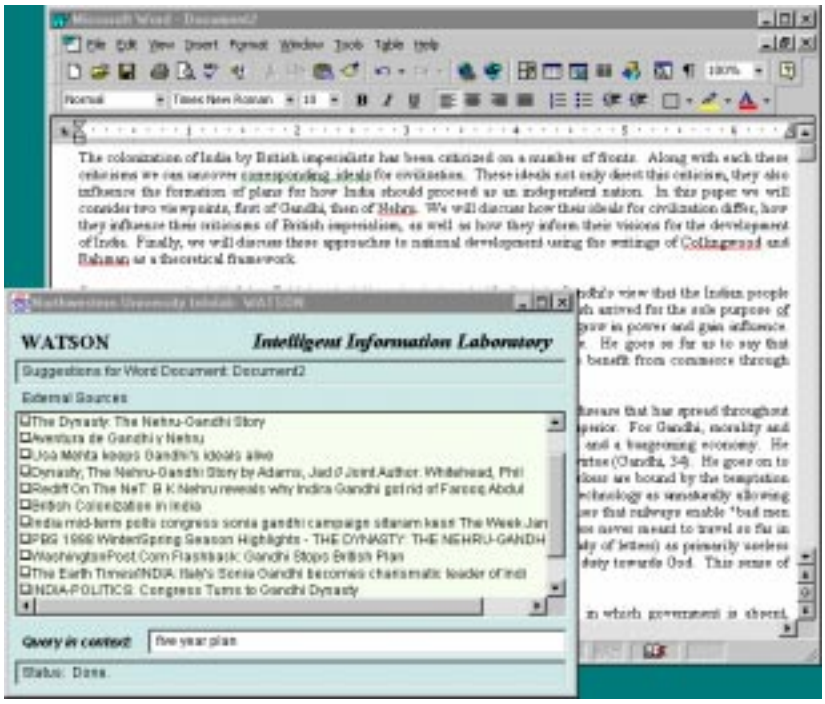

Figure 1: Watson suggesting information sources to assist in a research paper.

tentially be increased without burden for the user. The goal of the SourceSelect project is to develop methods for automatically identifying relevant information and satisfying the information needs.

SourceSelect

SourceSelect bridges the gap between a representation of the type of information relevant to the user's task, as generated by Watson, and information sources on the Internet. The aim is a scalable approach that can improve focus while requiring minimal knowledge to be coded. Consequently, we have begun by investigating the use of IR methods to form the association between queries and sources. The choice of sources is based whenever possible on easily accessible information that does not require representing the focuses of the search engines by hand.

Our method divides the search engines used by Watson into two groups, general and specific. Every focused search engine has a list of keywords associated with it, keywords gathered from the META tags on the search engine's main page. A small percentage of search engines do not have keywords in META tags; their keyword lists are constructed manually. The system can currently access six specialized search engines for various topics: CNN, CNNfn, Indiana University, the India engine Khoj, HumorSearch, and ESPN.

When a query is generated by the Watson engine, the SourceSelect uses a vector-space retrieval algorithm (Salton & McGill 1983) to go against the keywords for each search engine, to find specialized search engines relevant to the query (recall that Watson's queries are processed to include terms associated with the task context). This identifies a set of search engines whose focuses are believed relevant to the query, based on a pre-set threshold for sufficient relevance. (This threshold has been set arbitrarily, but we plan to investigate the effects of tuning.) The query is then sent to the selected specialized search engines in addition to the general search engines. For some search engines, the length of the query is reduced to the first few terms to improve retrieval performance. When results are returned from these search engines, they are sent back to the Watson engine for clustering and display

When the selected search engines are especially appropriate, this method can markedly improve the quality of the results generated for a query. For example, while browsing a page on `www.cbs.com` concerning the Dow Jones industrial average crossing the 10,000 mark, the suggestions in Table 1 were generated by standard Watson and Watson with SourceSelect (page titles are shown). The original version of Watson found some sites that relate to financial news, but the results were not very useful for someone with an interest in the Dow. With SourceSelect, the keywords Watson generated for the paper matched with keywords for the CNN financial search engine, and better results were produced.

Two key questions for this approach are whether the selection of specialized search engines will improve re-

Standard Watson	Watson with SourceSelect
WDBJ 7 news at 6 for 08/11/96	Technology Stocks Slip In Lackluster Trading
The 6 O'Clock Report, Wednesday, 8/5/98	When a fund company is publicly traded - Mar. 19, 1999
85 Documents about 'Dog bites & Stats'	Dow manages slight gain in early morning trading
http://www.io.com/ nuka/Text/kpnuka981029.txt	Toon Inn
Log from the hatching of . . . Kereneth's Clutch Ista...	Dow slides 31.13 in jittery trading
Factors Influencing Media Coverage of Business Crises	CNNfn - Dow Squeezes out gain - Nov. 4, 1997
	CNNfn - Dow breaks . . . losing streak - June 17, 1996
	Dow closes up 337.17 in record gain on busiest . . . day ever
	Tech Stocks Solid As Dow And Nasdaq Gain

Table 1: Example of Watson results with and without source selection.

sults for queries in their context area, and whether possible erroneous selection of specialized search engines will degrade performance for queries that are not in their content area. Informal trials are encouraging and we are now designing experiments to test these two questions.

Issues

Issues for automatic source selection include how to identify the user's information needs, how to select sources relevant to those needs, and how to access and exploit the information they provide. We discuss each of these in turn.

Identifying needed information

A key goal of IMAs is to automatically provide users with the right information, rather than forcing them to interrupt their tasks as they notice needs for information and try to satisfy those needs through manual searching. Achieving this goal depends on the system being able to determine what information is relevant to the current goals, without directly querying the user. In principle, abductive plan recognition could be used to explain the user's actions and anticipate information needs. In practice, however, there are many reasons this is not possible: it is too difficult to generate high-level explanations for user behavior, processing cost is too high, too much background knowledge is required, and too many explanations are possible for the observed behaviors.

The Watson approach is to use limited task knowledge, at the level of how particular applications are used and how to infer content information likely to be relevant, to guide its description of relevant content. For example, Watson's knowledge includes that headings in documents are likely to be important. Based on this knowledge, it describes the important content of a document by generating a term vector that gives greater weights to terms in headings. Thus content-relevance is used as an easier-to-compute proxy for task-relevance.

An issue to explore is whether it is worthwhile to preserve the context independently of a query describing information needs within that context. In this approach, the context alone would be used to select specialized search engines to then be presented with the query that assumes that context.

Source characterization

Our initial method for describing the focuses of specific search engines relies on the keywords selected by search engine developers to describe them. These tags provide a reasonable first pass to characterizations, but there is no guarantee that these tags will be accurate. (In some cases the inaccuracies are intentional, as search engines add popular tags merely to increase the chance that the tags for their search engines will match queries presented to other search engines, to increase their traffic.) We plan to explore other methods for characterizing information sources, such as generating term vectors directly from crawling site contents for accessible repositories. We also plan to investigate methods for more flexible matching of page descriptions, such as using a hierarchy to provide more flexible matching for related terms.

Engine-Specific Query Generation

Being able to select specialized information sources raises interesting questions about how to transform general queries into queries that exploit the contextual focus provided by a specialized search engine. When generating a query for a general-purpose search engine such as AltaVista, much of the query content is needed to disambiguate the required context. Once a context is established by the specialized source, that information is no longer necessary. Some search engines automatically AND the terms in queries as their default processing mode (e.g., ESPN), making it possible that the additional terms included for disambiguation will prevent useful information from being retrieved. (For www.humorsearch.com, which has a very small database, queries with more than two terms appear to seldom retrieve *any* results.) In general, being able to access specialized information sources raises interesting questions of how to tailor queries to those sources, in light of both the information needed and the characteristics of the sources themselves.

Parser Selection and Wrapper Generation

Accessing specialized search engines requires having mechanisms for extracting the information that they return and making it available in a useful form. This corresponds to the well-known problem of wrapper generation. SourceSelect relies on hand-coded wrappers to access its information sources, but ideally would exploit either a standard set of wrappers to allow semi-automatic selection or wrapper learning methods (e.g., (Kushmerick, Doorenbos, & Weld 1997)) to facilitate the addition of new sources. Effective methods for wrapper generation are one precondition for automatic addition of new information sources.

Collating Results

A final issue is how to merge the results of multiple specialized sources. SourceSelect currently relies on heuristic clustering algorithms in Watson to group results. These algorithms use information such as the titles of pages and the structure of URLs to decide when two pages are similar. For specialized information sources, these heuristics could be augmented with heuristics that also consider the implicit context provided by the sources of the information themselves.

Perspective

The basic Watson system addresses task-relevant focusing by automatically generating queries relevant to content areas associated with the task. The addition of SourceSelect adds task-based focusing for selecting where the query is sent. The premises of this approach contrast dramatically with those of a search engine such as Google (http://www.google.com), in which the goal of a search is to find a "consensus" answer. In our approach, the goal of a search is to find the answer most relevant to a specific information-seeking context, and the use of specialized resources helps assure the relevance of the result to that context.

Surprisingly little work has been done on source selection. The most notable example, a previous version of SavvySearch (Dreilinger & Howe 1997) kept track of how well search engines handled past queries, and used vector-space retrieval to match the current query to a search engine that has previously done well with similar queries. ProFusion (Gauch & Wang 1996) used a handbuilt knowledge hierarchy to categorize queries and select relevant search engines. More recently, an agent-based learning system was added to ProFusion to manipulate each engine's place in the hierarchy based on past searches (Fan & Gauch 1999).

Older systems, like Metacrawler (Selberg & Etzioni 1995) use only general search engines and send the query to all of them. Bandwidth constraints limit the number of search engines that can be queried. The current incarnation of SavvySearch (http://www.savvysearch.com) now appears to use this approach as well.

The Internet Sleuth (http://www.isleuth.com) is a search engine that indexes other specialized search engines. It allows the user to effectively perform a source-selection algorithm by hand.

Apple's Sherlock (http://www.apple.com/sherlock) allows the user to select the search engines that will be queried. This approach puts the burden of source selection entirely on the user. The user is forced to remember which search engines give the most relevant results for each type of query he may want to use.

The GlOSS (Gravano, Garcia-Molina, & Tomasic 1994) system obtains the index from each of its information sources, and combines these indices to form a meta-index, which is used for source selection. The drawback of this approach is that all of the information sources must cooperate by providing their indices in order for the meta-index to be built.

EMIR (Kulyukin 1999) maintains positive and negative keyword vectors for each of its information sources. Like GlOSS, it needs the cooperation of the information sources to maintain an accurate representation of their contents.

Most of these systems, use general-purpose search engines for their information sources. While general-purpose search engines provide the broadest coverage, focused search engines can have a much greater concentration of relevant links within their subject area. When Watson's contextual information is added to basic source selection, focused search engines appear to provide better results than general search engines.

Conclusion

SourceSelect augments Watson's just-in-time retrieval framework with the capability to choose specialized information sources related to the current context. The goal is to leverage off existing information resources to automatically provide the user with task-relevant information. The current version of SourceSelect matches term vector descriptions of the content area of interest to descriptions from the tags of specialized search engines to select sources expected to be relevant to those content areas, queries those sources, and forwards those results to Watson for presentation to the user. Initial tests have been encouraging; next steps include addition of other specialized search engines, formal evaluation, and exploration of alternative methods for describing task-relevant content and selecting information sources.

References

Budzik, J., and Hammond, K. 1999. Watson: a just-in-time information environment. In *AAAI Workshop on Intelligent Information Systems.* In Press.

Budzik, J.; Hammond, K.; Marlow, C.; and Scheinkman, A. 1998. Anticipating information needs: Everyday applications as interfaces to internet information resources. In *Proceedings of the 1998*

World Conference on the WWW, Internet, and Intranet.

Dreilinger, D., and Howe, A. 1997. Experiences with selecting search engines using meta-search. *ACM Transactions on Information Systems* 15(3).

Fan, Y., and Gauch, S. 1999. Adaptive agents for information gathering from multiple distributed information sources. In *Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace*. AAAI Press.

Gauch, S., and Wang, G. 1996. Information fusion with ProFusion. In *WebNet '96: The First World Conference of the Web Society*, 174–179.

Gravano, L.; Garcia-Molina, H.; and Tomasic, A. 1994. Precision and recall of gloss estimators for database discovery. In *Proceedings of the third international Conference on Parllel and Distributed Information Systems (PDIS '94)*.

Kulyukin, V. 1999. Application-embedded retrieval from distributed free-text collections. In *Proceedings of the Sixteenth National Conference on Artifical Intelligence*. AAAI Press. In press.

Kushmerick, N.; Doorenbos, R.; and Weld, D. 1997. Wrapper induction for information extraction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann.

Salton, G., and McGill, M. 1983. *Introduction to modern information retrieval*. New York: McGraw-Hill.

Selberg, E., and Etzioni, O. 1995. Multi-service search and comparison using the metacrawler. In *Proceedings of the Fourth World Wide Web Conference*, 195–208.

Understanding Web Augmentation

Oscar Díaz

ONEKIN Research Group, University of the Basque Country (UPV/EHU),
San Sebastián, Spain
oscar.diaz@ehu.es

Introduction

The increasing volume of content and actions on the web, combined with the growing number of "digital natives", anticipate a growing desire of more sophisticated ways of controlling the Web experience. Webies 2.0 do no longer take the web as it is but imagine fancy ways of customizing the web for their own purposes. So far, mashups are the forerunner exponent of this tendency where consumers (companies and laymen alike) come up with new applications by synergistically combining third-party resources. This presentation moves the focus to another approach: *"Web Augmentation"* (WA). Rather than creating a new application, WA builds on top of the rendering of an existing website. In some sense, WA is to the Web what Augmented Reality is to the physical world: layering relevant content/layout/navigation over the existing Web to customize the user experience. Unlike mashups, the purpose for WA is not so much coming up with a new application, but framing the new development within the Web experience of an existing website. Since this is achieved by third parties in a non-intrusive way, WA is a client-side technology: extensive use of *JavaScript (JS)* using browser weavers (e.g. *Greasemonkey*) or plugs-in. Rationales for WA include:

- addressing long-tail requirements. Minority usage patterns might not be worth considering in the general release of a website but even so, be catered for as plugs-in to be deployed in an individual basis. An example is *A Bit Better Remember-The-Milk (RTM)* [3]. This plug-in improves the navigation experience of the *RTM* website through a side navigation bar that speeds up specific ways to access to-do tasks. The most of users will be satisfy by the RTM website but this does not prevent long-tail demands from being served by augmenting the RTM website,
- affordance. A company might increase the affordance of its services by transparently embedding its offerings as parts of someone else's website. An example is the *Skype* add-on, [4] a plug-in that turns any phone number found in a web page into a button that launches *Skype* to call that number. The security company AVG provides another example. Its plugin *LinkScanner* [2] scans search results from *Google, Yahoo!* or *Bing*, and places a safety rating next to each recovered link that informs about the trustworthiness of the site,

M. Grossniklaus and M. Wimmer (Eds.): ICWE 2012 Workshops, LNCS 7703, pp. 79–80, 2012.

– end-user customization. Skilful users might also adapt their frequently-visited websites to their specific needs. *BookBurro* [1] is a case in point. This is a plug-in for price comparison at Amazon. *BookBurro*'s developer is a frequent *Amazon* buyer that likes to check other online bookshops before the purchase. From this perspective, WA departs from more traditional personalization scenarios where the website itself either caters for the adaptation (i.e. Web Personalization) or provides the means for register users to configure their Web experience (i.e. Web Customization).

We anticipate a quick eclosion of these "augmentations" as Web users demand more sophisticated ways of controlling the Web experience. As an evidence of this impulse, a repositoy for augmentation scripts, *www.userscripts.org,* holds over 85,000 scripts. Despite these figures, WA is still an art without a clear definition of its aim, good practices or development guidelines.

The presentation advocates for a more rigurous WA development by proposing (1) a set of good practices, (2) an architectural pattern for WA, and (3), a case for an agile approach to WA development. A non-trivial Wikipedia augmentation is used as a running example. In so doing, we hope to pave the way towards empowering users and organization alike with principles and methodologies that make the Web a truly customizable space.

Acknowledgements. Thanks are due to Cristobal Arellano who participates in the birth of these ideas. This work is co-supported by the Spanish Ministry of Education, and the European Social Fund under contract TIN2011-23839 *(Scriptongue).*

References

1. Andrews, B.: Book Burro (2010), `https://addons.mozilla.org/addon/book-burro/`
2. AVG: AVG LinkScanner - How it Works (2010), `http://linkscanner.avg.com/ww.sals-how-it-works.html`
3. Paprotsky, A.: A Bit Better RTM (2009), `https://addons.mozilla.org/addon/a-bit-better-rtm/`
4. Skype: Skype button in Internet Explorer or Firefox toolbar (2005), `http://www.skype.com/intl/en/support/user-guides/toolbar?lang=en`

Bulletin of the Technical Committee on

Data Engineering

September 2000 Vol. 23 No. 3 **IEEE Computer Society**

Letters

Special Issue on Next-Generation Web Search

Announcements and Notices

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems.

The Data Engineering Bulletin web page is http://www.research.microsoft.com/research/db/debull.

Distribution

IEEE Computer Society
1730 Massachusetts Avenue
Washington, D.C. 20036-1992
(202) 371-1013
nschoultz@computer.org

Letter from the Editor-in-Chief

Please Read: Bulletin Formats

In an effort to simplify the preparation of the Bulletin, and to reduce disk storage on our server, I would like to eliminate some of the versions of the Bulletin that are available on this web site (and via ftp). Specifically, I propose to *eliminate the draft versions* of the Bulletin. These are the versions in which many figures are not included. The purpose of the draft version was to increase the probability that something could be successfully printed, even if the figure could not. I have heard no complaints from anyone about their ability to display or print the Bulletin. The current issue (September, 2000) will include draft versions on the web site. But I intend to discontinue them for subsequent issues. Indeed, I intend to remove all draft versions from the Bulletin web site. If you believe you need draft versions, please send me mail at `lomet@microsoft.com`. I am willing to re-consider this should there be sufficient concern expressed about this. But please, only send me comments if you actually depend upon the availability of the draft versions.

About the Current Issue

For those of us (most of us) who are regularly "on-line", web searching has become an everyday part of our lives. And it is surely not only the technical community, but people everywhere who are benefitting from web search technology. Indeed, an important part of the usefulness of the web lies exactly in the ability to find information that very frequently derives from many sources from all over the world. So the impact of effective web search has been enormous, and will continue to grow in importance over time, as even more people access ever more information made available at web sites around the world.

Of course, the challenging nature of web search and its commercial opportunities have not escaped the attention of both the research and the industrial database communities. Indeed, the distinction between research and industrial communities has all but disappeared in many cases as researchers frequently join startups in the exploitation and improvement of web searching. The field is moving very rapidly, and those with database skills are playing a substantial role in the development of the field. After all, search is one of the things that database folks have been doing for over thirty years.

Luis Gravano is the editor for the current issue. He has assembled an issue that includes articles describing work going on in both research labs and industrial enterprises. Much of this work is an effort to move beyond "vanilla" key word search by considering "meta" information derived from the way that information is organized on the web, both for the semantic relevance of the result and for assessing the quality of the information. Luis has done a fine job in assembling this very interesting and timely collection of articles on web search, and I thank him for this. I am confident that readers will find this issue to be both interesting and useful.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

Web search engines have made substantial progress in helping users find information effectively. Nowadays a few select search engines answer tens of millions of queries a day over roughly a billion web pages, taking on average under a second to process a query. At the same time, overall user satisfaction for some of these engines is high. However, web search technology is far from mature, and there is still plenty of room for improvement. More specifically, users are often overwhelmed with query results that include many irrelevant pages. An important on-going challenge for search engines is to guess what users are after given only very few words as input. Also, current search engines ignore the often valuable contents of search-only text databases available on the web, since crawlers cannot download and index the contents of such databases. Recent studies have estimated the size of this "hidden web" to be several orders of magnitude that of the "crawlable web." On a related note, web pages sometimes include relatively structured information that would be most useful if extracted and made available via specialized interfaces that exploited this structure. The papers in this issue of the Bulletin address the research challenges in building next-generation web search engines with intuitive, expressive interfaces for all kinds of web-accessible information.

The first paper in this issue, by Monika Henzinger (Google, Inc.), surveys recent work on exploiting the link structure of the web to improve the quality of search results. Before the web existed, traditional information retrieval research studied for decades how to identify the text documents that are topically most relevant to a given query. A key assumption underlying this research was that the collection of available text documents was reasonably uniform in quality, value, and intended audience. This assumption does not hold over the web, hence the importance of adding other factors (e.g., popularity) when producing query results. The second paper, by Alberto Mendelzon (U. of Toronto) and Davood Rafiei (U. of Alberta), shows a novel use of the topology of the web to compute the set of topics on which a given web page has high "reputation." The third paper, by William Cohen, Andrew McCallum, and Dallan Quass (WhizBang! Labs–Research), discusses how to build topic-specific search engines that identify, extract, and integrate structured information present on web pages. Such specialized search engines can then let users ask complex queries that are appropriate for the domain and topic of the engines. The fourth paper, by Steve Lawrence (NEC Research), argues that next-generation search engines should consider the user's "context" to choose what resources (e.g., specialized search engines) to return and exploit. The fifth paper, by Jamie Callan (Carnegie Mellon U.), describes distributed search models for search engines, where the evaluation of queries potentially spans a number of autonomous, searchable databases, rather than a single large, monolithic, centralized collection as with current search engines. Finally, the paper by Marti Hearst (U. of California, Berkeley) discusses how to build effective interfaces for the different flavors of web resources.

Web search is a unique research area in its potential to directly impact the way hundreds of millions of users interact with information. I hope you will find the six papers in this issue of the Bulletin to be a good sample of the exciting research that is being pursued in this area in academia, research labs, and start-up companies.

Luis Gravano
Columbia University

Link Analysis in Web Information Retrieval

Monika Henzinger
Google Incorporated
Mountain View, California
monika@google.com

Abstract

The analysis of the hyperlink structure of the web has led to significant improvements in web information retrieval. This survey describes two successful link analysis algorithms and the state-of-the art of the field.

1 Introduction

The goal of information retrieval is to find all documents relevant for a user query in a collection of documents. Decades of research in information retrieval were successful in developing and refining techniques that are solely word-based (see e.g., [2]). With the advent of the web new sources of information became available, one of them being the *hyperlinks* between documents and records of user behavior. To be precise, *hypertexts* (i.e., collections of documents connected by hyperlinks) have existed and have been studied for a long time. What was new was the large number of hyperlinks created by independent individuals. Hyperlinks provide a valuable source of information for web information retrieval as we will show in this article. This area of information retrieval is commonly called *link analysis*.

Why would one expect hyperlinks to be useful? A hyperlink is a reference of a web page B that is contained in a web page A. When the hyperlink is clicked on in a web browser, the browser displays page B. This functionality alone is not helpful for web information retrieval. However, the way hyperlinks are typically used by authors of web pages can give them valuable information content. Typically, authors create links because they think they will be useful for the readers of the pages. Thus, links are usually either navigational aids that, for example, bring the reader back to the homepage of the site, or links that point to pages whose content augments the content of the current page. The second kind of links tend to point to high-quality pages that might be on the same topic as the page containing the link.

Based on this motivation, link analysis makes the following simplifying assumptions:

- A link from page A to page B is a recommendation of page B by the author of page A.
- If page A and page B are connected by a link the probability that they are on the same topic is higher than if they are not connected.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Link analysis has been used successfully for deciding which web pages to add to the collection of documents (i.e., which pages to *crawl*), and how to order the documents matching a user query (i.e., how to *rank* pages). It has also been used to categorize web pages, to find pages that are related to given pages, to find duplicated web sites, and various other problems related to web information retrieval.

The idea of studying "referrals" is, however, not new. A subfield of classical information retrieval, called bibliometrics, analyzed citations (see, e.g., [19, 14, 29, 15]). The field of sociometry developed algorithms [20, 25] very similar to the PageRank and HITS algorithms described below. Some link analysis algorithms can also be seen as collaborative filtering algorithms: each link represents an opinion and the goal is to mine the set of opinions to improve the answers to individuals.

This paper is structured as follows. We first discuss graph representations for the web (Section 2). In Section 3 we discuss two types of connectivity-based ranking schemata: a *query-independent* approach, where a score measuring the intrinsic quality of a page is assigned to each page without a specific user query, and a *query-dependent* approach, where a score measuring the quality and the relevance of a page to a given user query is assigned to some of the pages. In Section 4 other uses of link analysis in web information retrieval are described.

2 A Graph Representation for the Web

In order to simplify the description of the algorithms below we first model the web as a graph. This can be done in various ways. Connectivity-based ranking techniques usually assume the most straightforward representation: The graph contains a node for each page u and there exists a directed edge (u, v) if and only if page u contains a hyperlink to page v. We call this directed graph the *link graph* G.

Some algorithms make use of the undirected *co-citation graph*: As before each page is represented by a node. Nodes u and v are connected by an undirected edge if and only if there exists a third node x linking to both u and v.

The link graph has been used for ranking, finding related pages, and various other problems. The co-citation graph has been used for finding related pages and categorizing pages.

3 Connectivity-Based Ranking

3.1 Query-Independent Connectivity-Based Ranking

In *query-independent* ranking a score is assigned to each page without a specific user query with the goal of measuring the intrinsic quality of a page. At query time this score is used with or without some query-dependent criteria to rank all documents matching the query.

The first assumption of connectivity based techniques immediately leads to a simple query-independent criterion: The larger the number of hyperlinks pointing to a page the better the page. The main drawback of this approach is that each link is equally important. It cannot distinguish between the quality of a page pointed to by a number of low-quality pages and the quality of a page that gets pointed to by the same number of high-quality pages. Obviously, it is therefore easy to make a page appear to be high-quality – just create many other pages that point to it.

To remedy this problem Brin and Page [5, 26] invented the PageRank measure. The PageRank of a page is computed by weighting each hyperlink proportionally to the quality of the page containing the hyperlink. To determine the quality of a referring page, they use its PageRank recursively. This leads to the following definition of the PageRank $R(p)$ of a page p:

$$R(p) = \epsilon/n + (1 - \epsilon) \cdot \sum_{(q,p) \in G} R(q)/outdegree(q),$$

where

- ϵ is a dampening factor usually set between 0.1 and 0.2;
- n is the number of nodes in G; and
- $outdegree(q)$ is the number of edges leaving page p, i.e., the number of hyperlinks on page q.

Alternatively, the PageRank can be defined to be the stationary distribution of the following infinite random walk $p_1, p_2, p_3, \ldots$, where each p_i is a node in G: Each node is equally likely to be the first node p_1. To determine node p_{i+1} with $i > 0$ a biased coin is flipped: With probability ϵ node p_{i+1} is chosen uniformly at random from all nodes in G, with probability $1 - \epsilon$ node p_{i+1} is chosen uniformly at random from all nodes q such that edge (p_i, q) exists in G.

The PageRank is the dominant eigenvector of the probability transition matrix of this random walk. This implies that when PageRank is computed iteratively using the above equation, the computation will eventually converge under some weak assumptions on the values in the probability transition matrix. No bounds are known on the number of iterations but in practice roughly 100 iterations suffice.

The PageRank measure works very well in distinguishing high-quality web pages from low-quality web pages and is used by the Google[1] search engine.

The PageRank algorithm assigns a score to each document independent of a specific query. This has the advantage that the link analysis is performed once and then can be used to rank all subsequent queries.

3.2 Query-Dependent Connectivity-Based Ranking

In *query-dependent* ranking a score measuring the quality and the relevance of a page to a given user query is assigned to some of the pages.

Carriere and Kazman [11] proposed an indegree-based ranking approach to combine link analysis with a user query. They build for each query a subgraph of the link graph G limited to pages on the query topic. More specifically, they use the following query-dependent *neighborhood graph*. A *start set* of documents matching the query is fetched from a search engine (say the top 200 matches). This set is augmented by its *neighborhood*, which is the set of documents that either point to or are pointed to by documents in the start set. Since the indegree of nodes can be very large, in practice a limited number of predecessors (say 50) of a document are included. The neighborhood graph is the subgraph of G induced by the documents in the start set and its neighborhood. This means that each such document is represented by a node u and there exists an edge between two nodes u and v in the neighborhood graph if and only if there is a hyperlink between them. The indegree-based approach then ranks the nodes by their indegree in the neighborhood graph. As discussed before this approach has the problem that each link counts an equal amount.

To address this problem, Kleinberg [21] invented the *HITS* algorithm. Given a user query, the algorithm first iteratively computes a *hub* score and an *authority* score for each node in the neighborhood graph[2]. The documents are then ranked by hub and authority scores, respectively.

Nodes, i.e., documents that have high authority scores are expected to have relevant content, whereas documents with high hub scores are expected to contain hyperlinks to relevant content. The intuition is as follows. A document which points to many others might be a good hub, and a document that many documents point to might be a good authority. Recursively, a document that points to many good authorities might be an even better hub, and similarly a document pointed to by many good hubs might be an even better authority. This leads to the following algorithm.

(1) Let N be the set of nodes in the neighborhood graph.

[1] `http://www.google.com/`

[2] In the HITS algorithm the neighborhood graph is slightly modified to exclude edges between nodes on the same host. The reason is that hyperlinks within the same host might be by the same author and hence might not be a recommendation.

(2) For every node n in N, let $H[n]$ be its hub score and
$A[n]$ its authority score.
(3) Initialize $H[n]$ to 1 for all n in N.
(4) While the vectors H and A have not converged:
(5) For all n in N, $A[n] := \sum_{(n',n)\in N} H[n']$
(6) For all n in N, $H[n] := \sum_{(n,n')\in N} A[n']$
(7) Normalize the H and A vectors.

Since this algorithm computes the dominant eigenvectors of two matrices, the H and A vectors will eventually converge, but no bound on the number of iterations is known. In practice, the vectors converge quickly.

Note that the algorithm does not claim to find *all* valuable pages for a query, since there may be some that have good content but have not been linked to by many authors or that do not belong to the neighborhood graph.

There are two types of problems with this approach: First, since it only considers a relatively small part of the web graph, adding edges to a few nodes can potentially change the resulting hubs and authority scores considerably. Thus it is easier for authors of web pages to manipulate the hubs and authority scores than it is to manipulate the PageRank score. See [23] for a more extensive discussion of this problem. A second problem is that if the neighborhood graph contains more pages on a topic different from the query, then it can happen that the top authority and hub pages are on this different topic. This problem was called *topic drift*. Various papers [7, 8, 4] suggest the use of edge weights and content analysis of either the documents or the anchor text to deal with these problems. In a user study [4] it was shown that this can considerably improve the quality of the results.

A recent paper by Lempel and Moran [23] gives anecdotal evidence that a variant of the indegree-based approach achieves better results than the HITS algorithm. They compute the stationary distribution of a random walk on an auxiliary graph. This corresponds to scaling the indegree of a node u in the link graph by the relative size of u's connected component in the co-citation graph and the number of edges in u's component in the auxiliary graph. Basically, each link is weighted and the quality of a page is the sum of the weights of the links pointing to it. However, more experimental work is needed to evaluate this approach.

3.3 Evaluation of Query-Dependent Rankings

Amento, Terveen, and Hill [1] evaluated different link-based ranking criteria on a graph similar to the neighborhood graph. They start from a seed-set of relevant pages for a given query and their goal is to rank them by quality using various criteria.

The seed-set has the property that no url in the seed-set is the prefix of another one. They consider these urls to be *root urls* of *sites*: all pages which contain the root url as prefix belong to the site of this root url. Then they perform a neighborhood expansion using link and text similarity heuristics and restricting the expansion to pages on the above sites. For their analysis they use either this graph or a *site graph*, where all pages on a site are collapsed to one node. Note that the set of nodes in the site graph is fully determined by the seed-set and the neighborhood expansion is used only to determine the edges in the site graph.

They use five link-based metrics (in-degree, out-degree, HITS authority score, HITS hub score, and PageRank) and some other metrics to rank the root urls by either using the score assigned to the root url (in the pages-based graph) or to the site (in the site graph). Interestingly, the ranking on the site graph outperformed the ranking on the pages-based graph. Furthermore, there is a large overlap and correlation in the rankings of the in-degree, HITS authority score, and PageRank metric and these three metrics perform roughly equally well. They also outperform the other metrics together with another simple metric that counts the number of pages on a site that belong to the graph.

Note, however, that they perform the PageRank computation on a small graph, while the PageRank computation described before was performed on the whole link graph and the resulting PageRank values will most likely differ considerably.

4 Other Uses of Link Analysis in Web Information Retrieval

Apart from ranking, link analysis can also be used for deciding which web pages to add to the collection of web pages, i.e., which pages to crawl. A *crawler* (or *robot* or *spider*) performs a traversal of the web graph with the goal of fetching high-quality pages. After fetching a page, it needs to decide which page out of the set of uncrawled pages to fetch next. One approach is to crawl the pages with highest number of links from the crawled pages first. Cho et al. propose to visit the pages in the order of PageRank [10].

Link analysis was also used for a search-by-example approach to searching: given one relevant page find pages related to it. Kleinberg [21] proposed using the HITS algorithm for this problem and Dean and Henzinger [12] show that both the HITS algorithm and a simple algorithm on the co-citation perform very well. The idea behind the latter algorithm is that frequent co-citation is a good indication of relatedness and thus the edges with high weight in the co-citation graph tend to connect nodes with are related.

Extensions of the HITS and PageRank approaches were used by Rafiei and Mendelzon to compute the reputation of a web page [27] and by Sarukkai to predict personalized web usage [28].

Almost completely mirrored web hosts cause problems for search engines: they waste space in the index data structure and might lead to duplicate results. Bharat et al. [3] showed that a combination of IP address analysis, URL pattern analysis, and link structure analysis can detect many near-mirrors. The idea is that near-mirrors exhibit as very similar link structure within the host as well as to the other hosts.

Chakrabarti et al. [9] made first steps towards using the link structure for web page categorization.

In [17, 18] PageRank-like random walks were performed on the web to sample web pages almost according to the PageRank distribution and the uniformly distribution, respectively. The goal was to compute various statistics on the web pages and to compare the quality, respectively the number, of the pages in the indices of various commercial search engines.

Buyukkokten et al. [6] and Ding et al. [13] classify web pages based on their geographical scope by analyzing the links that point to the pages.

5 Conclusions

The main use of link analysis is currently in ranking query results. Other areas were link analysis has been shown to be useful are crawling, finding related pages, computing web page reputations and geographic scope, prediction link usage, finding mirrored host, categorizing web pages, and computing statistics of web pages and of search engines.

However, research of the hyperlink structure of the web is just at its beginning and a much deeper understanding needs to be gained.

References

[1] B. Amento, L. Terveen, and W. Hill. Does authority mean quality? Predicting expert quality ratings of web documents. In *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'00)*, pages 296–303.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

[3] K. Bharat, A. Z. Broder, J. Dean, and M. Henzinger. A comparison of techniques to find mirrored hosts on the World Wide Web. *Workshop on Organizing Web Space (WOWS)* in conjunction with *ACM Digital Library '99*. To appear in the *Journal of the American Society for Information Science*, 2000.

[4] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in hyperlinked environments. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pages 111–104.

[5] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the Seventh International World Wide Web Conference* 1998, pages 107–117.

[6] O. Buyukkokten, J. Cho, H. García-Molina, L. Gravano, and N. Shivakumar. Exploiting geographical location information of Web pages. *Proc. of the ACM SIGMOD Workshop on the Web and Databases (WebDB'99)*, 1999.

[7] S. Chakrabarti, B. Dom, D. Gibson, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. In *ACM–SIGIR'98 Post-Conference Workshop on Hypertext Information Retrieval for the Web*.

[8] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proceedings of the Seventh International World Wide Web Conference* 1998, pages 65–74.

[9] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998, pages 307–318.

[10] J. Cho, H. García-Molina, and L. Page. Efficient crawling through URL ordering. In *Proceedings of the Seventh International World Wide Web Conference* 1998, pages 161–172.

[11] J. Carriere and R. Kazman. Webquery: Searching and visualizing the web through connectivity. In *Proceedings of the Sixth International World Wide Web Conference* 1997, pages 701–711.

[12] J. Dean and M. R. Henzinger. Finding related Web pages in the World Wide Web. In *Proceedings of the 8th International World Wide Web Conference* 1998, pages 389–401.

[13] J. Ding, L. Gravano, and N. Shivakumar. Computing geographical scopes of Web resources. *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, 2000.

[14] E. Garfield. Citation analysis as a tool in journal evaluation. *Science*, 178, 1972.

[15] E. Garfield. *Citation Indexing.* ISI Press, 1979.

[16] T. Haveliwala. Efficient computation of PageRank. Technical Report 1999-31, Stanford University, 1999.

[17] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring search engine quality using random walks on the Web. In *Proceedings of the 8th International World Wide Web Conference* 1999, pages 213–225.

[18] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform URL sampling. In *Proceedings of the Ninth International World Wide Web Conference* 2000, pages 295–308.

[19] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14, 1963.

[20] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39-43, March 1953.

[21] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.

[22] J. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The Web as a graph: Measurements, models and methods. Invited survey at the *International Conference on Combinatorics and Computing*, 1999.

[23] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proceedings of the Ninth International World Wide Web Conference* 2000, pages 387–401.

[24] D. S. Modha and W. S. Spangler. Clustering hypertext with applications to Web searching. *Proceedings of the ACM Hypertext 2000 Conference, San Antonio, TX*, 2000. Also appears as IBM Research Report RJ 10160 (95035), October 1999.

[25] M. S. Mizruchi, P. Mariolis, M. Schwartz, and B. Mintz. Techniques for disaggregating centrality scores in social networks. In N. B. Tuma, editor, *Sociological Methodology*, pages 26–48. Jossey-Bass, San Francisco, 1986.

[26] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the Web. *Stanford Digital Library Technologies*, Working Paper 1999-0120, 1998.

[27] D. Rafiei, and A. Mendelzon. What is this page known for? Computing Web page reputations. In *Proceedings of the Ninth International World Wide Web Conference* 2000, pages 823–836.

[28] R. Sarukkai. Link prediction and path analysis using Markov chains. In *Proceedings of the Ninth International World Wide Web Conference* 2000, pages 377–386.

[29] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *J. Amer. Soc. Info. Sci.*, 24, 1973.

What do the Neighbours Think? Computing Web Page Reputations

Alberto O. Mendelzon
Department of Computer Science
University of Toronto
mendel@cs.toronto.edu

Davood Rafiei
Department of Computing Science
University of Alberta
drafiei@cs.ualberta.ca

Abstract

The textual content of the Web enriched with the hyperlink structure surrounding it can be a useful source of information for querying and searching. This paper presents a search process where the input is the URL of a page, and the output is a ranked set of topics on which the page has a reputation. For example, if the input is www.gamelan.com, then a possible output is "Java." We describe a simple formulation of the notion of reputation of a page on a topic, and report some experiences in the use of this formulation.

1 Introduction

The idea of exploiting the "reputation" of a Web page when searching has attracted research attention recently and even been incorporated into some search engines [16, 6, 11, 2, 3]. The idea is that pages with good reputations should be given preferential treatment when reporting the results of a search; and that link structure can be mined to extract such reputation measures, on the assumption that a link from page a to page b is, to some degree, an endorsement of the contents of b by the creator of a. The question that needs to be answered to use page reputation in Web search is: given a topic, what pages have the highest reputation on this topic? We consider a different question in this paper: given a page (or a Web site), on what topics is this page considered an authority by the Web community?

There are many potential applications for such computations. For example, a company may wish to know how its Web site is categorized by other pages that point to it, for several reasons: to assess its popularity, to check whether it is projecting the right image, or to detect problems signaled by unflattering links. Statistics of web access, such as the unique monthly visitor counts maintained by Web ranking services, are notoriously controversial [18] and do not provide insight on the topics that a Web site is perceived to be relevant to. A link-based reputation measuring service could be used not only by the site owners, but by anyone who needs to evaluate a site before using it as a source of information, or before transacting business with it.

Another example is the use of the reputation measure of a Web site listing a researcher's publications to help assess the impact of the researcher's work, with the obvious caveats against depending too heavily on any one measure of impact– caveats that, for that matter, also apply to more traditional methods such as print or online citation indexes.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

There are difficulties in formalizing the notion of "reputation" effectively. The assumption that links are endorsements suggests that the number of incoming links of a page indicates its reputation. But in practice, links represent a wide variety of relationships such as navigation, subsumption, relatedness, refutation, justification, etc. In addition, we are interested not just in the overall reputation of a page, but in its reputation on specific topics.

In this paper, we describe a search process where the input is the URL of a page, and the output is a ranked set of *topics* on which the page has a reputation. (For our purposes, a topic is simply a term or a phrase, an admittedly simplistic definition.) For example, if the input is `www.informatik.uni-trier.de/~ley/db`, then among the outputs we would like to see with high rank topics such as "DBLP Bibliography", "database systems," etc. We present a simple formulation of the notion of reputation with the following two goals in mind: (1) it must be easy to compute in the setting of the Web; (2) it must be effective in measuring the reputations of pages. We point out that this formulation is a rough approximation to a more complex measure, based on random walk models of Web browsing behaviour. Finally, we report some test results on the effectiveness of our computations.

1.1 Related Work

Recent analyses of the linkage structure of the Web suggest that hyperlinks between pages often represent relevance [16, 6] or endorse some authority [11, 2, 3].

In a method incorporated into the Google search engine, Brin and Page [3] compute the importance of a Web page as the sum of the importance of its incoming links. The computation simulates the behaviour of a "random surfer" who either selects an outgoing link uniformly at random, or jumps to a new page chosen uniformly at random from the entire collection of pages. The PageRank of a page corresponds to the number of visits the "random surfer" makes to the page.

Kleinberg [11] proposes an algorithm that, given a topic, finds pages that are considered authorities on that topic. The algorithm, known as HITS, is based on the hypothesis that for broad topics, authority is conferred by a set of *hub pages*, which are recursively defined as a set of pages with a large number of links to many relevant authorities.

In earlier work [17], we developed two formulations of the notion of reputation based on random walk models of simple Web browsing behaviours. The first model, based on the idea of one-level influence propagation, generalizes PageRank; the main difference is that the ranking is performed with respect to a specific topic instead of computing a universal rank for each page. The second model is a probabilistic formulation of a model similar to Hubs and Authorities; this formulation allows us to invert the search process, i.e., given the URL of a page, we can find the topics the page is an authority on. For differences between HITS and probabilistic approaches, see the work by Lempel and Moran [15].

The notion of adjusting link weights for HITS was studied by Chakrabarti et al. [4] and Bharat and Henzinger [2]. Based on the hub-and-authority structure of a community, Kumar et al. [13] show that a large number of such communities can be identified from their signatures in the form of complete bipartite subgraphs of the Web. Dean and Henzinger [6] suggest algorithms to find related pages of a given page solely based on the linkage structure around the page. Finally, Henzinger et al. [9, 10] use random walks on the Web to do URL sampling and also to measure the quality of pages stored in an index.

2 Our Approach

Given a page, we want to identify a ranked list of topics the page has a reputation on. This requires a ranking function and a collection (of pages and topics) over which the ranks can be computed.

We first define two ratios that relate a page *p* and a topic *t*. The *penetration* of page *p* on topic *t*, $P_p(t)$, is the fraction of pages on topic *t* that point to page *p*. (For our purposes, a page is *on* topic *t* simply when it contains

the term or phrase *t*.) The *focus* of page *p* on topic *t*, $F_t(p)$, is the fraction of pages pointing to *p* that are on topic *t*. That is, if $I(p,t)$ is the number of pages that contain *t* and point to *p*, $In(p)$ is the number of pages that point to *p*, and *N(t)* is the number of pages that contain *t*, then

$$P_p(t) = I(p,t)/N(t)$$

and

$$F_t(p) = I(p,t)/In(p).$$

Note that these quantities can be interpreted as conditional probabilities: $P_p(T)$ is the conditional probability that a page points to *p*, given that it contains *t*, and $F_t(p)$ is the conditional probability that a page contains *t*, given that it points to *p*.

Using one of these ratios as a measure of the reputation of *p* on *t* is problematic. For example, $P_p(t)$ may overestimate the reputation on any topic of pages that have large in-degrees, while $F_t(p)$ may overestimate the reputation of those whose incoming links are narrowly concentrated on topic *t*. It is more appropriate to consider, not just the conditional probability that a page points to *p* given that it contains *t*, but how much larger (or smaller) is this probability than the unconditional probability that an arbitrary page points to *p*. Note that the latter probability is given by $L(p) = In(p)/N_w$, where N_w is the number of pages on the web. Let us define the *reputation measure* of *p* on *t*, $RM(p,t)$, as

$$RM(p,t) = (P_p(t) - L(p))/L(p).$$

We could equally define $RM(p,t)$ in terms of $F_t(p)$, letting $M(t) = N(t)/N_w$ be the probability that an arbitrary page contains term *t*:

$$RM(p,t) = (F_t(p) - M(t))/M(t) = (P_p(t) - L(p))/L(p).$$

If we now define $LM(t,p)$ as the probability that a page contains term *t* and points to page *p*, it is easily seen that

$$RM(p,t) = [LM(t,p)/(L(p) \times M(t))] - 1.$$

That is, $RM(p,t)$ measures how far from independent are the events "a page contains *t*" and "a page points to *p*." Brin *et al.* [1] propose a similar measure, called a *dependence rule*, as an alternative to association rules (which use confidence ratios similar to $P_p(t)$ and $F_t(p)$), and show how to use standard statistical tests to evaluate its significance; we do not pursue the latter topic in this paper.

Given a search engine that can compute estimates of $I(p,t)$, $N(t)$ and N_w, $RM(P,t)$ can be readily estimated as

$$RM(p,t) = (N_w I(p,t)/N(t)In(p)) - 1.$$

We next show that the proposed ranking provides both an effective and easy to compute reputation measure.

2.1 Ranking Effectiveness

We now provide some justifications on the effectiveness of the proposed ranking. When a page is created, it has no incoming links (except possibly some links from the same site, which we ignore in our computation). As other users become aware of the page, based on their judgments of the content of the page and its relevance to their topics of interest, they start including links to the page within the pages they create or maintain. After a while, if a large fraction of pages on a specific topic point to the page, it is natural to expect that the page has secured a reputation on that topic.

A similar interesting phenomenon can be seen in link navigation. Users frequently search the Web by alternating between the two modes of (1) searching for pages that contain some terms (using a search engine), and (2) following outgoing links from those pages. If a large fraction of pages on a specific topic points to a page,

the page will be most likely visited by the Web surfers searching for pages on that topic. Specifically, for a given page *p* and topic *t*, $RM(p, t)$ is, to some degree, a rough approximation to the number of visits made to page *p* by a "random surfer" who wanders the Web by following the links and searching for pages on topic *t* (see [17] for details).

2.2 Rank Computations

Consider the rank computation in the simple case where both a page *p* and a word or phrase *t* are given. We can use a search engine to estimate the values of $In(p)$, $I(p, t)$ and $N(t)$, and then plug these estimates in to compute $RM(p, t)$. For example, we can estimate $In(p)$, $I(p, t)$ and $N(t)$ by respectively sending queries "+link:p," "+link:p +t" and "+t" to AltaVista (*www.altavista.com*) and retrieving the counts returned by the engine. The value of N_w, a constant which doesn't affect the ordering, can be estimated by the number of pages in the search engine collection; this is often publicly announced (e.g., [14]).

However, we are often interested in the case where the topics that a page has a reputation on are not known in advance. Thus the problem is: how to compute for a given page *p* the set of topics *t* such that $RM(p, t)$ is highest? A solution is to compute $RM(p, t)$ for every word or phrase that appears in page p. Although this is easy to compute, it is not good enough because, as pointed out in the literature, often a page is an authority on some term that is not mentioned in the page. For example, the IBM Almaden Research Center (*www.almaden.ibm.com*) has a reputation on "data mining," but this phrase does not appear anywhere in its home page. [1]

Another solution is to compute $RM(p, t)$ for every possible word or phrase that appears in a page that points to *p*. (We call such pages "incoming links" of *p*.) In general, this is infeasible in the setting of the Web since there can be tens or hundreds of thousands of incoming links, and examining all those links is a cumbersome process.

We now describe the solution adopted by TOPIC [19], a prototype developed at the University of Toronto for computing Web page reputations. Given a page, the system compiles a set of incoming links of the page. This is currently done using AltaVista and Lycos, but it can be equally done using other search engines. The size of the result set is limited by the maximum number of entries returned by the search engine; search engines often return no more than 1000 entries. Clearly the accuracy of the ranks will depend on the fraction of incoming links returned. Then, for each incoming link, the system extracts words and phrases from the "snippet" returned by the search engine, rather than the page itself. This avoids the additional overhead of downloading the page, under the expectation that the snippet of a page, to some degree, is representative of the topic of the page.

There are other issues which need to be dealt with. First, links within the same site are often created for navigation purposes; as mentioned above, these links are ignored. Second, it is quite likely to find one or more near-duplicate copies of the same document in the search engine collection, even though search engines usually try to avoid storing duplicates. To address this problem, duplicate snippets are removed. Third, stop words such as "the," "for," "in," etc. and rare terms such as "BBAAA" usually convey no specific meanings (for the purpose of computing page reputations) and are removed.

3 Examples

In this section, we report our experiences with TOPIC, a prototype that uses the proposed method for reputation measurements. The input to the prototype is the URL of a page, and the output is a ranked list of topics. The default value for the number of links to download is 300, but the user can change it to a smaller number (to get better speed) or to a larger number (to get better precision). The user can enter an optional term or phrase, in which case the reputation of the page is measured only on that particular topic, and the result is displayed within a list of top 10 authorities identified by Google [8] for the same topic, providing a comparative ranking. The

[1] Disclaimer: all the examples are current as of 00/08/28, but things change fast on the Web.

default search engine for downloading the incoming links and estimating the query counts is AltaVista, but the user can change it to another engine (currently Lycos being the only alternative).

3.1 Known Authoritative Pages

For our first experiment, we selected the home pages of a set of U.S. database research groups, whose reputations we knew, from the DBLP bibliography [5]. [2] As shown in Figure 1, the results look quite reasonable. Note the high reputation of the *Microsoft Research* home page on the phrase "Data Engineering Bulletin," due to the fact that the site hosts the online version of this Bulletin.

URL : www-db.research.bell-labs.com 192 links examined (out of 193 available) **Topics**: Database Systems, Data Mining, ACM, Databases, Computer Science
URL : www.research.microsoft.com/research/db 212 links examined (out of 349 available) **Topics**: Technical Committee on Data Engineering, IEEE Data, Active Databases, Data Engineering Bulletin, Database Research, SIGMOD, SIGMOD Record, DBLP, VLDB, Database Systems
URL : www.almaden.ibm.com 999 links examined (out of 7815 available) **Topics**: IBM Almaden Research Center, Search Engines, Data Mining, Microscopy, Visualization
URL : www-db.stanford.edu 1000 links examined (out of 5637 available) **Topics**: Database research, Data Warehousing, Database Systems, Data Mining, Stanford
URL : db.cs.berkeley.edu 73 links examined (out of 130 available) **Topics**: Computer Science, Berkeley, Database, Research
URL : www.db.ucsd.edu 43 links examined (out of 78 available) **Topics**: XML, Database, Research, Project, Information

Figure 1: Selected database research group home pages and their reputations

3.2 Unregulated Web Sites

For our second experiment, we selected the home pages of a number of Canadian departments of Computer Science. The main characteristic of these sites is that they are unregulated, in the sense that users store any documents they desire in their own pages. Our goal was to find out how a site is perceived overall, without singling out specific pages stored on the site.

The results, as shown in Figure 2, can be surprising. The Computer Science Department at the University of Toronto has a high reputation on "Russian history" and "travel" mainly because a Russian graduate student of the department has put online a large collection of resources on Russia, and many pages on Russia link to it. The high reputation on "hockey" is due to a former student who used to play on the Canadian national women's hockey team. The Department of Computer Science at the University of British Columbia has a high reputation on "periodic table" because the site keeps an online version of the periodic table of chemical elements. The site also has a high reputation on "Anime" and "Manga," Japanese animation and comic art, because a staff member has put online a collection of pages on the subject and many other pages link to this collection. The reputation

[2] Disclaimer: the selection is arbitrary and not intended to be a complete list of groups of high reputation.

of the Department of Computing Science at the University of Alberta on "virtual reality" and "chess" and the reputation of the Department of Computing Science at Simon Fraser University on "data mining" and "reasoning" are to be expected. The high reputation of the CS Department at Simon Fraser University on "whales" is due to a 3D animation project being carried out on whales.

The results reported here, even though some are surprising, turn out to be consistent with other data once we examine the pages in question. For example, the Russian page at the University of Toronto reported over one million hits since 1997 for a period of two years; the number of unique visits (visits with different host names) to the page was slightly over 333,000. Similarly, Andria Hunter's hockey page, for the period of two weeks starting from July 19, 2000 showed 2,700 average daily visits. The total number of visits since the creation of the page in March 1995 was about 1.8 million.

URL : www.cs.toronto.edu 1000 links examined (out of 8400 available) **Topics**: Russian History, Neural, Travel, Hockey
URL : www.cs.ualberta.ca 1000 links examined (out of 10557 available) **Topics**: University of Alberta, Virtual Reality, Language, Chess, Artificial
URL : www.cs.ubc.ca 999 links examined (out of 17958 available) **Topics**: Confocal, Periodic Table, Anime, Computer Science, Manga, Mathematics
URL : www.cs.sfu.ca 963 links examined (out of 2055 available) **Topics**: Whales, Simon Fraser University, Data Mining, Reasoning

Figure 2: Selected Computer Science Department home pages and their reputations

3.3 News Sites

So far, we have only ranked topics for a given site. For this purpose it does not matter whether we use $RM(p, t)$ or the penetration $P_p(t)$, since they produce the same ordering of topics for a fixed p. In this experiment, we evaluated a set of sites, all of them news providers, on a predetermined set of topics. For a fixed topic t, ranking a set of sites by their value of $RM(p, t)$ amounts to ordering them by their focus $F_t(p)$. Since we are interested in comparing both within and across sites, we show in Figure 3, both the penetration and focus for each combination of site and topic, revealing interesting patterns. For example, *CNN* has the largest penetration of any site on every topic, but relatively low focus, showing that it is well-known on all the topics but not specifically known for any single one. On the other hand, *wired.com*, while ranking a close second in penetration to *CNN* on "technology," has substantially higher focus on this topic than any other site.

3.4 Personal Home Pages

In another experiment, we selected a set of personal home pages and used our system to find the reputation topics for each page. We expected this to describe in some way the reputation of the owner of the page. The results, as shown in Figure 4, can be revealing, but need to be interpreted with some care. Tim Berners-Lee's reputation on the "History of the Internet," Don Knuth's fame on "TeX" and "Latex," Jeff Ullman's reputation on "database systems" and "programming languages" and Jim Gray's reputation on "database," "research" and "systems" are to be expected. The Comprehensive TeX Archive Network (CTAN) is frequently cocited with Don Knuth's home page mainly within TeX information pages. Alberto Mendelzon's high reputation on "data warehousing" and

	CNN (www.cnn.com)	BBC (www.bbc.co.uk)	ABC (www.abcnews.go.com)	Wired.com
International News	[0.0277 , 0.0170]	[0.0121 , 0.0201]	[0.0021 , 0.0279]	[0.0057 , 0.0090]
Weather	[0.0096 , 0.2228]	[0.0029 , 0.1845]	[0.0005 , 0.2338]	[0.0013 , 0.0744]
Sports	[0.0043 , 0.1724]	[0.0012 , 0.1265]	[0.0003 , 0.2531]	[0.0017 , 0.1698]
Entertainment	[0.0074 , 0.1632]	[0.0049 , 0.2913]	[0.0003 , 0.1516]	[0.0023 , 0.1294]
Travel	[0.0040 , 0.1421]	[0.0016 , 0.1548]	[0.0003 , 0.2264]	[0.0012 , 0.1082]
Technology	[0.0041 , 0.1957]	[0.0011 , 0.1390]	[0.0002 , 0.2234]	[0.0038 , 0.4560]
Business	[0.0034 , 0.2831]	[0.0018 , 0.3946]	[0.0002 , 0.3399]	[0.0022 , 0.4635]

Figure 3: [P , F] ranks of news provider sites on a set of topics

"OLAP," on the other hand, is due to an online research bibliography he maintains on these topics, and not to any merits of his own.

URL : www.w3.org/People/Berners-Lee 789 links examined (out of 1334 available) **Topics**: Tim Berners-Lee, History Of The Internet, Hypertext, Pioneers, Brief, W3C
URL : www-cs-faculty.stanford.edu/~ knuth 1000 links examined (out of 1543 available) **Topics**: Don Knuth, TeX Users, LaTex, Linux, CTAN, Donald, Computer Science
URL : www-db.stanford.edu/~ ullman 294 links examined (out of 423 available) **Topics**: Ullman, Database Management Systems, Database Systems, Database Design, Data Mining, Programming Languages
URL : www.research.microsoft.com/~ gray 57 links examined (out of 74 available) **Topics**: Database, Research, Systems, Information
URL : www.cs.toronto.edu/~ mendel 161 links examined (out of 162 available) **Topics**: Data Warehousing, OLAP, Data Mining, Bibliography, Computer Science, Database, Research

Figure 4: Selected personal home pages and their reputations

4 Conclusion

We have presented a method for evaluating the reputation of a page which is both easy to compute and, according to our preliminary tests, effective. There are some limitations to our method that should be mentioned. First, we don't exploit relationships among terms such as synonyms, generalization, specialization, etc. Second, our computation is affected by the fraction of incoming links returned by search engines and the unpredictable order in which they are returned: this affects both the choice of relevant topics and the estimation of the ratios. We are currently working on refinements of the method to overcome these limitations, as well as on more systematic evaluation of the method's effectiveness.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council Canada and Communications Information Technology Ontario.

References

[1] S. Brin, R. Motwani, and C. Silverstein, Beyond market baskets: generalizing association rules to dependence rules. *Data Mining and Knowledge Discovery* 2(1), pages 39–68, 1998.

[2] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in hyperlinked environments. In *Proc. of the ACM SIGIR Conference*, pages 104–111, 1998.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proc. of the WWW7 Conference*, pages 107–117, Brisbane, Australia, April 1998. Elsevier Science.

[4] S. Chakrabarti, B. Dom, P. Raghavan, S. Rajagopalan, D. Gibson, and J. Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *Proc. of the WWW7 Conference*, Brisbane, Australia, April 1998. Elsevier Science.

[5] DBLP Bibliography. www.informatik.uni-trier.de/~ley/db.

[6] J. Dean and M. R. Henzinger. Finding related pages on the Web. In *Proc. of the WWW8 Conference*, pages 389–401, Toronto, Canada, May 1999. Elsevier Science.

[7] D. Florescu, A. Levy, and A. Mendelzon. Database techniques for the World Wide Web : a survey. *ACM SIGMOD Record*, 27(3):59–74, September 1998.

[8] Google. www.google.com.

[9] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. Measuring index quality using random walks on the Web. In *Proc. of the WWW8 Conference*, pages 213–225, Toronto, Canada, May 1999. Elsevier Science.

[10] M. R. Henzinger, A. Heydon, M. Mitzenmacher, and M. Najork. On near-uniform url sampling. In *Proc. of the WWW9 Conference*, pages 295–308, Amsterdam, May 2000. Elsevier Science.

[11] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, January 1998.

[12] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the Web. In *Proc. of the VLDB Conference*, pages 639–650, September 1999.

[13] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. In *Proc. of the WWW8 Conference*, pages 403–415, Toronto, May 1999. Elsevier Science.

[14] S. Lawrence and C.L. Giles. Accessibility of information on the Web. *Nature*, 400:107–109, 1999.

[15] R. Lempel and S. Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In *Proc. of the WWW9 Conference*, pages 387–401, Amsterdam, May 2000. Elsevier Science.

[16] Netscape Communications Corporation. What's related. Web page, www.netscape.com/escapes/related/faq.html.

[17] D. Rafiei and A. O. Mendelzon. What is this page known for? Computing Web page reputations. In *Proc. of the WWW9 Conference*, pages 823–835, Amsterdam, May 2000. Elsevier Science.

[18] Jon Swartz. Net rankings vex dot-coms. *USA Today*, June 2000. www.usatoday.com/life/cyber/invest/in794.htm.

[19] TOPIC. www.cs.toronto.edu/db/topic.

Learning to Understand the Web

William Cohen[‡]
wcohen@whizbang.com

Andrew McCallum[§]
mccallum@whizbang.com
WhizBang! Labs–Research

Dallan Quass[¶]
dquass@whizbang.com

Abstract

In a traditional information retrieval system, it is assumed that queries can be posed about any topic. In reality, a large fraction of web queries are posed about a relatively small number of topics, like products, entertainment, current events, and so on. One way of exploiting this sort of regularity in web search is to build, from the information found on the web, comprehensive databases about specific topics. An appropriate interface to such a database can support complex structured queries which are impossible to answer with traditional topic-independent query methods. Here we discuss three case studies for this "data-centric" approach to web search. A common theme in this discussion is the need for very robust methods for finding relevant information, extracting data from pages, and integrating information taken from multiple sources, and the importance of statistical learning methods as a tool for creating such robust methods.

1 Introduction

In a traditional information retrieval system, it is assumed that queries can be posed about any topic. In reality, a large fraction of web queries are posed about a relatively small number of topics, like products, entertainment, current events, and so on. One way of exploiting this sort of regularity in web search is to build, from the information found on the web, comprehensive databases about specific topics. An appropriate interface to such a database can support complex structured queries (*e.g.*, "bed and breakfast units within one mile of the beach in Hawaii costing less than $150 per night") which are impossible to answer with a topic-independent query method.

In this paper, we will motivate this "data-centric" approach to web search, and then discuss three case studies: namely WHIRL, CORA, and FLIPDOG. In a typical WHIRL application, data from a few dozen web sites is merged together to form a database; however, the extracted data is generally "dirty," or approximate, in several key respects. WHIRL uses textual similarity metrics from information retrieval to approximately answer structured queries to this "approximate" database of web information. CORA and FLIPDOG extract information about specific types of entities (research papers and job postings, respectively) from thousands of different web sites.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[‡]Mailing address: WhizBang! Labs, East, 4616 Henry Street, Pittsburgh PA 15213
[§]Mailing address: WhizBang! Labs, East, 4616 Henry Street, Pittsburgh PA 15213
[¶]Mailing address: WhizBang! Labs, 3210 N. Canyon Road, Suite 300, Provo Utah 84604

Both CORA and FLIPDOG rely heavily on machine learning methods to discover relevant information sources, extract information about entities, and organize entities into categories. A common theme in our discussion of these systems is the need for very robust methods for finding and classifying relevant information, extracting data from pages, and integrating information taken from multiple sources, and the importance of statistical learning methods as a tool for creating such robust methods.

2 Data-Centric Web Search

Traditional web search methods have certain fundamental limitations. Topic-independent search methods rely heavily on matching terms in a user's natural language query with terms appearing in a document. This approach is broadly applicable, but complex information needs (*e.g.*, the sample query of the introduction) cannot be expressed easily in a topic-independent system. Another limitation is that returning a single list of documents, ranked by estimated relevance, may not satisfy a user's query. For instance, the answer to the query "find technical job postings in companies whose stock has increased 50% or more in the last two years" may require combining information from two different sources—one which contains information about stock prices, and one which contains job postings.

We believe that such complex information needs are real (even if they are not often formulated by users of existing search engines). One approach to handling these queries is to compile the textual information on the web into some formalism that supports structured queries—for example, a relational database. In this "data-centric" approach to web information access, one begins by focusing on a particular topic: this is an important first step, as this makes it possible to build a database with deep, semantically meaningful schema. One then constructs a "topic-specific search engine": an information access system that allows access to all the information on the web that is relevant to a particular topic. We will assume here that a "topic-specific search engine" includes some sort of facility for answering structured queries, such as a relational database interface, but other organizational schemes (like a Yahoo-style topic hierarchy for documents) may also be used.

For example, suppose that a topic-specific search engine for travel were to be constructed. If I were interested in vacationing in Hawaii, I might browse the topic hierarchy to "Lodging," at which point I could either continue to drill down into the topic hierarchy, or I could narrow my search more quickly through a relational-style query over the information in lodging. Suppose that "type of lodging," "location," and "room rate" had been extracted from each lodging document and stored in a relational database. A search form could be created at the lodging node of the topic hierarchy that would allow me to query over these fields. I could specify the criteria "bed and breakfast units within one mile of the beach in Hawaii costing less than $150 per night." The result would be a list of bed and breakfast units that matched my search criteria, including their names, addresses, and phone numbers (also extracted from the documents), along with links to the actual source documents.

3 Building Topic-Specific Search Engines

While a topic-specific search engine is conceptually quite simple, building one is not. The essential problem involves classifying web pages relevant to the topic and extracting structured information that can be stored in a database. Web information on a topic may be generated by thousands of different content providers in thousands of different formats. This information is continually changing and growing, and the scale of the web is such that only an automatic or nearly-automatic approach can hope to maintain a database for a reasonably broad topic. Furthermore, most of the web data available is presented as simple text—a format easy for people to generate and understand, but extremely difficult for computers to process. These factors make it extremely difficult to automatically convert web information into a format suitable for storing in a single coherent database.

One possible solution to this problem is to require that providers generate information in some database-compatible format, such as XML. Unfortunately, while XML is a great way to share information coming out of

databases, XML is far more difficult for end users to generate than simple textual documents. We cannot expect end-users to consistently tag their documents with the relevant pieces of information and to conform to standard DTDs. Nor is it reasonable to expect that programmers will mount a large-scale effort to convert the world's existing text to XML data. Therefore, although we believe that XML will be useful for exchanging information, we do *not* believe that most information will be created originally in XML. For the foreseeable future, web information will be presented primarily as text.

We believe that the most feasible approach for collecting web information into topic-specific search systems is to automatically extract information from the web using learned procedures. For example, one type of learning system might allow users to create topic hierarchies "by example." A user would create a topic hierarchy and place a few example documents into that hierarchy. The system would then gather additional documents from the web similar to these examples, and present them to the user for verification, essentially asking "is this the kind of thing you are interested in?" After such a dialog, the system would learn to automatically classify documents into the hierarchy. Analogously, a user could show examples of the relational-database fields that should be extracted from a document. After training, the system would learn to extract these fields automatically and populate a topic-specific relational database.

Although it is not necessary for such a tool to incorporate learning or programming-by-example, there are several reasons for believing that learning is the "right" approach to this problem. One advantage is that no programming is necessary—all the user need do is provide and verify examples. This greatly lowers the cost of developing a topic-specific search engine. Another advantage is that learned classifiers and extractors are often more robust than hand-written rules, in the sense that they are less likely to break down on unusual inputs.

Below we will present several examples of data-centric web search systems, and discuss the ways in which learning and other robust, statistical methods are used. First, however, we will give some additional background on the main techniques used in a topic-specific search engine: focused crawling, text classification, and information extraction.

4 Background and Related Work

Research in the areas of focused crawling, text classification, and information extraction has been on-going for many years. Early research methods suffered from low accuracy or required a significant amount of hand-tuning. Only recently have methods been developed that achieve sufficient accuracy without extensive hand-tuning as to make topic-specific search engines feasible.

As the Web continues to grow exponentially, the idea of crawling the entire Web on a regular basis becomes less and less feasible. Since only a relatively small portion of the Web is relevant to a topic-specific search engine, it can use the notion of focused crawling (*e.g.*, [4, 3, 6, 19]) to find those pages quickly, and bypass most pages that are not related to the topic. Crawling the web in general involves gathering up the links on each visited page and putting them into a queue of links to be followed. Focused crawling reorders the links in the queue as to their predicted likelihood to lead to pages that are relevant to a particular topic. By following high-likelihood links first, pages that are relevant to a particular topic are found more quickly. Furthermore, links leading to irrelevant pages tend to fall to the bottom of the queue, and can be ignored once the crawler has determined that the rate of finding new relevant pages from following links has fallen below some given threshold.

Text classification is used in topic-specific search engines in at least two areas. First, it is used during crawling to classify web pages as to whether they are relevant to the given topic. Second, it can be used to classify relevant web pages or content extracted from web pages into a Yahoo-style topic hierarchy. Much research has been done in the area of text classification (*e.g.*, [7, 14]). The general idea is to first convert the text to a multidimensional vector representation, where dimensions correspond to words in the text, then to use machine-learning or statistical techniques (*e.g.*, decision trees, naive Bayes) to classify the vectors in the multidimensional space. The hypertext structure of the Web provides an additional advantage for web page classification that is not avail-

able for classifying text in general: One can use the text in links leading to the page and also the classification of nearby pages to make classification of a particular web page more accurate than if classification were performed on the text of the page alone.

The key to creating a topic-specific search engine is to extract values for specific fields from the web pages, storing the values in a database so that structured queries can be performed over the extracted information. A primary venue for research in information extraction has been the Message Understanding Conference (MUC) series. The MUC conferences were a series of contests where researchers would build working systems based upon their research and test their systems on real-world tasks. One set of tests involved extracting "named entities" such as locations, company names, or person names from flat text files such as newspaper articles. Early systems were typically built using a large set of hand-coded extraction rules. Later, some of the systems were built using machine-learning techniques (*e.g.*, [18]).

In comparison to extracting information from flat text files, it is possible to get increased accuracy when extracting information from web page by taking advantage of the HTML tag structure. One common approach to extracting information from web pages is to write site-specific "wrappers" that extract information based upon regularities of the HTML tag structure that is typically present in a single web site (*e.g.*, [10, 12]). For example, if a bed and breakfast web site listed all their bed and breakfast units in a list, it would be possible to extract information for each of those units by simply extracting the text following each "`<LI>`" tag. The disadvantage of this approach is that as web sites change over time, the tags the wrappers are dependent upon tend to change as well, requiring the wrappers to be rewritten.

Another method for extracting information from the Web is to use a bootstrapping approach (*e.g.*, [1, 2]). In this approach one begins with a small relation and searches the Web for additional tuples to add to the relation. New tuples are added to the relation using a bootstrapping technique, where new tuples are added if they appear on web pages in the same contexts as existing tuples in the relation.

5 Three Case Studies

Below we review two research prototypes and a commercial system that use the methods discussed above as well as additional research methods to create topic-specific search engines.

5.1 A Research Paper Search Engine: CORA

The CORA system [16, 15] automatically spiders, classifies and extracts computer science research papers from the Web. The papers in CORA are organized into a taxonomy with 75 leaves, and various fields such as author and title are extracted from each paper. Additionally, bibliographic information is extracted from each paper, allowing bibliometric analysis to be performed. It currently contains over 50,000 papers and is publically available at *www.cora.whizbang.com*.

The creation and maintenance of CORA relies heavily on artificial intelligence and machine learning techniques. The tasks can be broken down into four components: spidering, extraction, reference matching and classification.

Research papers are gathered from the web using an efficient, topic-directed spider based on reinforcement-learning. The spider uses the words in the context of a hyperlink to assign it an "expected future discounted reward," and follows the high-reward links with higher priority. Training data for the language model that makes the assignment is easily obtained by exhaustively spidering a few training sites, and counting the minimum number of hops from each hyperlink to a target page. Experiments show that our directed spider is three times more efficient than a spider based on breadth-first search, and also more efficient than other smart spiders that do not explicitly model future reward.

After spidering, the papers' titles, authors, and references are automatically extracted using hidden Markov models—a type of probabilistic finite state machine often used in speech recognition. Certain states are associated

Figure 1: Typical travel-oriented web data

with different database fields (such as *author* or *title*), and after calculating the most likely state path using the Viterbi algorithm, the word emissions associated with those particular states are said to be associated with their associated fields. Training data consists of a combination of hand-labeled data and data from large bibliography files found on the web. After learning the state-transition structure and the parameters from this training data, automatic extraction achieves over 90% accuracy.

Next, the extracted references are matched against the papers, so that the complete citation graph is built. This is done efficiently by a two-stage clustering process that uses two different distance metrics—first a cheap, approximate distance metric based on an inverted index, then an expensive, detailed distance metric based on a tuned string-edit distance [17]. The quadratic-time string-edit distance is only performed on the small subset of reference-pairs that are within some distance threshold according to the cheap distance metric. Once the citation graph is complete, we run a bibliometric analysis based on principal components analysis to automatically find "seminal" and "survey" papers.

Finally, the papers are automatically categorized into the topic hierarchy using probabilistic text classification. A small number of human-provided keywords, a large amount of unlabeled data, and a statistical technique for taking advantage of the hierarchy called "empirical Bayes" are all combined in a Bayesian classifier that provides near-human accuracy.

The CORA system can be adapted to a new domain by labeling the additional examples needed to learn a new set of classifiers—in fact, a second technical-paper search engine for statistics research papers was created in this way after just two days work. NEC's CiteSeer system [9] is another topic-specific search engine that is more specifically geared to research papers, but uses less machine learning.

5.2 Information Integration Systems and WHIRL

One set of data-centric web systems are "information integration systems" (*e.g.*, [13, 8, 11]), which collect into a single database the information from several heterogeneous, database-like web sites. As an example, consider the sample query above, and consider the web pages shown in Figure 1 (the second web page is one example of several reviews stored on the same site). A typical information integration system might extract information from these pages, converting the first web page into a relation of the form `nearBeach(hotel)` and the second web site into a relation `perNight(hotel,cost)`. The sample query above might be answered by joining these two relations and then filtering the result by cost.

One of the problems in doing information integration is that many database operations are very sensitive to errors in extracted data. As an example, if automatic, learned methods are used to extract data from the web pages above, it is quite possible that the `nearBeach` relation will contain the tuple ⟨`''Leie Away Inn''`⟩ and the `perNight` relation will contain the tuple ⟨`''Leie Away Guest House'',$125`⟩. (It is even pos-

sible that the `perNight` relation will contain a less-useful tuple, such as ⟨`''Review: The Leie Away Guest House'',$125`⟩.) This inconsistency means that neither version of this name will appear in the join of two relations.

The WHIRL integration system [5] is based on a novel representation scheme, which allows more extracted information to be stored as raw text. By making use of robust similarity metrics for text developed in the information retrieval community, many database-style operations can be approximated on this representation, even if the data is somewhat misaligned.

For instance, one might use the following WHIRL query to find inexpensive housing near the beach:

SELECT nearBeach.$*$, perNight.$*$
FROM nearBeach, perNight
WHERE nearBeach.hotel SIM perNight.hotel AND
perNight.cost $\leq$ 150

The output of this query will be k tuples from the cross-product of `perNight` and `nearBeach` that have the most similar `hotel` fields, subject to the constraint that `perNight.cost` is less than $150; thus the join of the relations `nearBeach` and `perNight` is approximated by finding tuples with similar hotel names. The number of tuples k returned by the query is fixed by the user, and similar approximations can be used for more complex queries.[1]

WHIRL views the process of finding the k best answers to a query as an optimization problem. For this query, WHIRL searches through the space of possible pairings of `nearBeach` and `perNight` tuples to find the pairings that maximize the given similarity condition ("`nearBeach.hotel SIM perNight.hotel`") using an artificial-intelligence optimization method called A$*$ search. To make this search efficient, WHIRL relies heavily on indexing methods used in information retrieval. In information retrieval, the similarity of two documents is a function of the set of *terms* those documents share, and the weight assigned to these shared terms. (For the purpose of this discussion, a term can be considered to be a single word.) For this query, WHIRL might use inverted indices to find `perNight` tuples that are guaranteed to share some "important" (highly-weighted) term in the `hotel` field of a candidate `nearBeach` tuple. More generally, appropriate use of inverted indices ensures that the most plausible pairings are explored early in the search.

Using robust versions of database operations eliminates the need for certain data-cleaning operations—in this example, for instance, it is not necessary to normalize the names of hotels. This often greatly simplifies the process of extracting data. Experimentally, WHIRL was used to provide interfaces to 50 different sites with approximately four man-months development time—a vast improvement over earlier systems—by relying on simple, approximate extraction schemes.

The sites included in the experiments with WHIRL were primarily on two topics—birds of North America, and educational computer games for children—and were primarily sites containing large amounts of data-like material. One site for the North American bird topic was `http://www2.math.sunysb.edu/~tony/birds` which contains hundreds of sound files containing bird calls; one site for the computer game domain was `http://www.kidsdomain.com` which contains hundreds of reviews of recent computer games. For each such site, a separate extraction routine was hand-written which spidered the site, retrieved the appropriate data, and converted it into WHIRL's internal format.

Because of WHIRL's robustness, many of these extraction routines could be quite simple (most were a single line in a special-purpose language). However, extracting data was still a bottleneck for WHIRL, because a routine had to be manually written and maintained for each site. We conjecture that coupling a WHIRL-like database system with CORA-style learned extraction methods would further reduce the cost of data collection.

[1] The current implementation of WHIRL supports a fairly large subset of SQL: specifically, any disjunction of conjunctive SQL queries has an analog in WHIRL.

5.3 A Commercial System: FLIPDOG.COM

Our organization, WhizBang! Labs, has developed a set of tools that facilitate the creation of topic-specific search engines. Recently, we have fielded a commercial system using these tools: FLIPDOG.COM, an on-line job board based on a database constructed by automatically extracting job postings directly from corporate Web pages. The FLIPDOG database was created primarily by applying general-purpose machine learning techniques—techniques that could well be used to extract other sorts of databases from the Web.

FLIPDOG contains more than 600,000 jobs gathered from over 50,000 different corporate web sites, making it the largest commercial job board on the Web. Operationally, the process of constructing the FLIPDOG database is much like the process used in CORA. Sites are automatically spidered to find pages that contain job postings. Individual job postings are then extracted from these pages, and augmented with automatically-extracted fields such as job title, job description, location, and application email address. Job postings are also organized into a taxonomy to facilitate browsing.

However, construction of the FLIPDOG database also required addressing a number of issues not solved by previous research prototypes. Unlike research papers, job postings are frequently accessible only by accessing forms, which means that the job-posting spider must be able to understand how to fill in these forms. The "location" field for a job posting is also harder to extract than, say, the title of a paper, because a job's location is often *not* explicitly mentioned in the job posting itself: *e.g.*, frequently a single Web page will name a location, and then list several jobs at that location.

FLIPDOG also embodies a new solution to the problem of errors in automatically-extracted data. In FLIPDOG, all automatically-made decisions about a job posting are associated with a "confidence"—a numeric measure of the system's certainty that the decision is correct—and human beings verify any low-confidence decisions. This means that the learning algorithms must not only provide accurate predictions, but accurate assessments of confidence. Adopting this approach means that FLIPDOG's job database is quite "clean," containing very few erroneous entries, but can still be refreshed on a weekly basis.

6 Summary

In general, constructing a topic-specific search engine requires solving many different problems, including identifying relevant information sources, extracting and classifying information, and integrating information taken from different sources. However, solutions to many of these problems are in hand, and have been implemented in various research prototypes and commercial systems.

We believe in the near future, toolkits consisting of various machine-learning-based techniques will make it much easier to classify and extract information from text. As the underlying technology matures, data-centric, topic-specific search engines will become more prevalent.

It is likely that many such search engines will be developed, each specializing in a different topic. Topic-specific engines are also likely to vary in their depth of coverage, with some systems electing to impose a rich schema on a smaller subset of the web, and others imposing a weak schema on a large subset of the web. Ultimately the vast majority of queries that focus on common topics will be answered by one of a few dozen general-purpose databases; and of the remaining, special-purpose queries, most will be answered by one of a few thousand more specialized databases, much like CORA. The information systems in wide use today will persist: however, traditional broad-coverage keyword-search based IR systems (like GOOGLE and ALTAVISTA) and highly-focused topic specific databases (like `imdb.com`, which contains only information about movies and TV) will simply be two ends of densely-populated spectrum of data-centric information systems, each providing a database-like view of a different part of the web.

References

[1] Eugene Agichtein and Luis Gravano. *Snowball:* extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, June 2000.

[2] Sergey Brin. Extracting patterns and relations from the World Wide Web. In *The World Wide Web and Databases, International Workshop Web'98*, Valencia, Spain, March 1998.

[3] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *Proceedings of The Eighth International World Wide Web Conference (WWW-99)*, Toronto, 1999.

[4] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the 7th World Wide Web Conference (WWW7)*, Brisbane, Australia, April 1998.

[5] William W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118:163–196, 2000.

[6] M. Diligenti, F. M. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB-2000)*, Cairo, Egypt, September 2000.

[7] Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the ACM Conference on Information and Knowledge Management (CIKM)*, November 1998.

[8] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The TSIMMIS approach to mediation: Data models and languages (extended abstract). In *Next Generation Information Technologies and Systems (NGITS-95)*, Naharia, Israel, November 1995.

[9] Lee Giles, Kurt Bollaker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, 1998.

[10] Ashish Gupta, Venky Harinarayan, and Anand Rajaraman. Virtual database technology. *SIGMOD Record*, 26(4):57—61, 1997.

[11] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, Madison, WI, 1998.

[12] Nicholas Kushmerick, Daniel S. Weld, and Robert Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Osaka, Japan, 1997.

[13] Alon Y. Levy and Rachel Pottinger. A scalable algorithm for answering queries using views. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB-2000)*, Cairo, Egypt, September 2000.

[14] Andrew McCallum and Kamal Nigam. A basic introduction to the two flavors of naive Bayes document classification. In *AAAI Workshop on Learning for Text Categorization*, 1998.

[15] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymour. Cora Research Paper Search. At `http://www.cora.whizbang.com`.

[16] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymour. Automating the construction of internet portals. To appear in *Information Retrieval*, 2000.

[17] Andrew McCallum, Kamal Nigam, and Lyle Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *KDD-2000: Proceedings of the Sixth International Conference on Knowledge Discovery and Data Mining*, 2000.

[18] Scott Miller, Michael Crystal, Heidi Fox, Lance Ramshaw, Richard Schwartz, Rebecca Stone, Ralph Weischedel, and the Annotation Group. Algorithms that learn to extract information BBN: description of the sift system as used for MUC-7. In *Proceedings of the 7th Message Understanding Conference (MUC-7)*, April 1998.

[19] Jason Rennie and Andrew McCallum. Using reinforcement learning to spider the web efficiently. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1999.

Context in Web Search

Steve Lawrence
NEC Research Institute
Princeton, New Jersey
http://www.neci.nec.com/~lawrence
lawrence@research.nj.nec.com

Abstract

Web search engines generally treat search requests in isolation. The results for a given query are identical, independent of the user, or the context in which the user made the request. Next-generation search engines will make increasing use of context information, either by using explicit or implicit context information from users, or by implementing additional functionality within restricted contexts. Greater use of context in web search may help increase competition and diversity on the web.

1 Introduction

As the web becomes more pervasive, it increasingly represents all areas of society. Information on the web is authored and organized by millions of different people, each with different backgrounds, knowledge, and expectations. In contrast to the databases used in traditional information retrieval systems, the web is far more diverse in terms of content and structure.

Current web search engines are similar in operation to traditional information retrieval systems [57] – they create an index of words within documents, and return a ranked list of documents in response to user queries. Web search engines are good at returning long lists of *relevant* documents for many user queries, and new methods are improving the ranking of search results [8, 10, 21, 36, 41]. However, few of the results returned by a search engine may be *valuable* to a user [6, 50]. Which documents are valuable depends on the context of the query – for example, the education, interests, and previous experience of a user, along with information about the current request. Is the user looking for a company that sells a given product, or technical details about the product? Is the user looking for a site they previously found, or new sites?

Search engines such as Google and FAST are making more information easily accessible than ever before and are widely used on the web. A GVU study showed that about 85% of people use search engines to locate information [31], and many search engines consistently rank among the top sites accessed on the web [48]. However, the major web search engines have significant limitations – they are often out-of-date, they only index a fraction of the publicly indexable web, they do not index documents with authentication requirements and many documents behind search forms, and they do not index sites equally [42, 43]. As more of the population goes online, and as more tasks are performed on the web, the need for better search services is becoming increasingly important.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Understanding the Context of Search Requests

Web search engines generally treat search requests in isolation. The results for a given query are identical, independent of the user, or the context in which the user made the request. Context information may be provided by the user in the form of keywords added to a query, for example a user looking for the homepage of an individual might add keywords such as "home" or "homepage" to the query. However, providing context in this form is difficult and limited. One way to add well-defined context information to a search request is for the search engine to specifically request such information.

2.1 Adding Explicit Context Information

The Inquirus 2 project at NEC Research Institute [29, 30] requests context information, currently in the form of a category of information desired. In addition to providing a keyword query, users choose a category such as "personal homepages", "research papers", or "general introductory information". Inquirus 2 is a metasearch engine that operates as a layer above regular search engines. Inquirus 2 takes a query plus context information, and attempts to use the context information to find relevant documents via regular web search engines. The context information is used to select the search engines to send queries to, to modify queries, and to select the ordering policy.

For example, a query for research papers about "machine learning" might send multiple queries to search engines. One of these queries might be transformed with the addition of keywords that improve precision for finding research papers (e.g., "abstract" and "references"). Another query might be identical to the original query, in case the transformations are not successful. Inquirus 2 has proven to be highly effective at improving the precision of search results within given categories. Recent research related to Inquirus 2 includes learning methods that automatically learn query modifications [18, 28].

2.2 Automatically Inferring Context Information

Inquirus 2 can greatly improve search precision, but requires the user to explicitly enter context information. What if search context could be automatically inferred? This is the goal of the Watson project [11, 12, 13]. Watson attempts to model the context of user information needs based on the content of documents being edited in Microsoft Word, or viewed in Internet Explorer. The documents that users are editing or browsing are analyzed with a heuristic term weighting algorithm, which aims to identify words that are indicative of the content of the documents. Information such as font size is also used to weight words. If a user enters an explicit query, Watson modifies the query based on the content of the documents being edited or viewed, and forwards the modified query to web search engines, thus automatically adding context information to the web search.

In addition to allowing explicit queries, Watson also operates in the background, continually looking for documents on the web related to documents that users are editing or viewing. This mode of operation is similar to the Remembrance Agent [54, 56]. The Remembrance Agent indexes specified files such as email messages and research papers, and continually searches for related documents while a user edits a document in the Emacs editor. Other related projects include: Margin Notes [55], which rewrites web pages to include links to related personal files; the Haystack project [1], which aims to create a community of interacting "haystacks" or personal information repositories; and Autonomy's Kenjin program (*www.kenjin.com*), which automatically suggests content from the web or local files, based on the documents a user is reading or editing. Also related are agents that learn user interest profiles for recommending web pages such as Fab [4], Letizia [47], WebWatcher [3], and Syskill and Webert [51].

2.3 Personalized Search

The next step is complete personalization of search – a search engine that knows all of your previous requests and interests, and uses that information to tailor results. Thus, a request for "Michael Jordan" may be able to rank links to the professor of computer science and statistics highly amongst links to the famous basketball player, for an individual with appropriate interests.

Such a personalized search engine could be either server or client-based. A server-based search engine like Google could keep track of a user's previous queries and selected documents, and use this information to infer user interests. For example, a user that often searches for computer science related material may have the homepage of the computer scientist ranked highly for the query "Michael Jordan", even if the user has never searched for "Michael Jordan" before.

A client-based personalized search service can keep track of all of the documents edited or viewed by a user, in order to obtain a better model of the user's interests. However, these services do not have local access to a large scale index of the web, which limits their functionality. For example, such a service could not rank the homepage of the computer scientist highly for the query "Michael Jordan", unless a search service returns the page within the maximum number of results that the client retrieves. The clients may modify queries to help retrieve documents related to a given context, however this is difficult for the entire interests of a user. Watson and Kenjin are examples of client-based personalized web search engines. Currently, Watson and Kenjin extract context information only from the current document that a user is editing or viewing.

With the cost of running a large scale search engine already very high, it is likely that server-based full-scale personalization is currently too expensive for the major web search engines. Most major search engines (Northern Light is an exception) do not even provide an alerting service that notifies users about new pages matching specific queries. However, advances in computer resources should make large scale server-based personalized search more feasible over time. Some Internet companies already devote a substantial amount of storage to individual users. For example, companies like DriveWay (*www.driveway.com*) and Xdrive (*www.xdrive.com*) offer up to 100Mb of free disk storage to each user.

One important problem with personalized search services is that users often expect consistency – they would like to receive the same results for the same queries, whereas a personalized search engine may return different results for the same query, both for different users, and also for the same user as the engine learns more about the user. Another very important issue, not addressed here, is that of privacy – many users want to limit the storage and use of personal information by search engines and other companies.

2.4 Guessing What the User Wants

An increasingly common technique on the web is guessing the context of user queries. The search engines Excite (*www.excite.com*), Lycos (*www.lycos.com*), Google (*www.google.com*), and Yahoo (*www.yahoo.com*) provide special functionality for certain kinds of queries. For example, queries to Excite and Lycos that match the name of an artist or company produce additional results that link directly to artist or company information. Yahoo recently added similar functionality, and provides specialized results for many different types of queries – e.g., stock symbols provide stock quotes and links to company information, and sports team names link to team and league information. Other examples for Yahoo include car models, celebrities, musicians, major cities, diseases and drug names, zodiac signs, dog breeds, airlines, stores, TV shows, and national parks. Google (*www.google.com*) identifies queries that look like a U.S. street address, and provides direct links to maps. Similarly, Google keeps track of recent news articles, and provides links to matching articles when found, effectively guessing that the user might be looking for news articles.

Rather than explicitly requiring the user to enter context information such as "I'm looking for a news article" or "I want a stock quote", this technique guesses when such contexts may be relevant. Users can relatively easily identify contexts of interest. This technique is limited to cases where potential contexts can be identified based

on the keyword query. Improved guessing of search contexts could be done by a personalized search engine. For example, the query "Michael Jordan" might return a link to a list of Prof. Michael Jordan's publications in a scientific database for a user interested in computer science, guessing that such a user may be looking for a list of publications by Prof. Jordan.

Clustering of search results, as performed by Northern Light for example, is related. Northern Light dynamically clusters search results into categories such as "current news" and "machine learning", and allows a user to narrow results to any of these categories.

3 Restricting the Context of Search Engines

Another way to add context into web search is to restrict the context of the search engine, i.e., to create specialized search engines for specific domains. Thousands of these search engines already exist (see *www.invisibleweb.com* and *www.completeplanet.com*). Many of these services provide similar functionality to regular web search engines, either for information that is on the publicly indexable web (only a fraction of which may be indexed by the regular search engines), or for information that is not available to regular search engines (e.g., the New York Times search engine). However, an increasing number of specialized search engines are appearing which provide functionality far beyond that provided by regular web search engines, within their specific domain.

3.1 Information Extraction and Domain-Specific Processing

ResearchIndex (also known as CiteSeer) [40, 44, 45] is a specialized search engine for scientific literature. ResearchIndex is a free public service (available at *www.researchindex.com*), and is the world's largest free full-text index of scientific literature, currently indexing over 300,000 articles containing over 3 million citations. It incorporates many features specific to scientific literature. For example, ResearchIndex automates the creation of citation indices for scientific literature, provides easy access to the context of citations to papers, and has specialized functionality for extracting information commonly found in research articles.

Other specialized search engines that do information extraction or domain-specific processing include DEADLINER [37], which parses conference and workshop information from the web, newsgroups and mailing lists; FlipDog (*www.flipdog.com*), which parses job information from employee sites; HPSearch (*http://hpsearch.uni-trier.de/hp/*), which indexes the homepages of computer scientists; and GeoSearch [14, 23], which uses information extraction and analysis of link sources in order to determine the geographical location and scope of web resources. Northern Light also provides a service called GeoSearch, however Northern Light's GeoSearch only attempts to extract addresses from web pages, and does not incorporate the concept of the geographical scope of a resource (for example, the New York Times is located in New York but is of interest in a larger geographical area, whereas a local New York newspaper may be of less interest outside New York).

Search engines like ResearchIndex, DEADLINER, FlipDog, HPSearch, and GeoSearch automatically extract information from web pages. Many methods have been proposed for such information extraction, see for example [2, 9, 20, 38, 39, 40, 58, 59].

3.2 Identifying Communities on the Web

Domain-specific search engines that target the publicly indexable web need a method of locating the subset of the web within their domain. Flake et al. [25] have recently shown that the link structure of the web self-organizes such that communities of highly related pages can be efficiently identified based purely on connectivity. A web *community* is defined as a collection of pages where each member has more links (in either direction) inside the community than outside of the community (the definition may be generalized to identify communities of various sizes and with varying levels of cohesiveness). This discovery is important because there is no central authority or process governing the formation of links on the web. The discovery allows identification of communities on

the web independent of, and unbiased by, the specific words used. An algorithm for efficient identification of these communities can be found in [25].

Several other methods for locating communities of related pages on the web have been proposed, see for example [7, 15, 16, 17, 22, 27, 36, 53].

3.3 Locating Specialized Search Engines

With thousands of specialized search engines, how do users locate those of interest to them? More importantly, perhaps, how many users will go to the effort of locating the best specialized search engines for their queries? Many queries that would be best served by specialized services are likely to be sent to the major web search engines because the overhead in locating a specialized engine are too great.

The existence of better methods for locating specialized search engines can help, and much research has been done in this area. Several methods of selecting search engines based on user queries have been proposed, for example GlOSS [33, 34] maintains word statistics on available databases, in order to estimate which databases are most useful for a given query. Related research includes [19, 24, 26, 32, 46, 49, 61, 62].

It would be of great benefit if the major web search engines attempted to direct users to the best specialized search engine where appropriate, however many of the search engines have incentives not to provide such a service. For example, they may prefer to maximize use of other services that they provide.

4 One Size Does Not Fit All, and May Limit Competition

Typical search engines can be viewed as "one size fits all" – all users receive the same responses for given queries. As argued earlier, this model may not optimally serve many queries, but are there larger implications?

An often stated benefit of the web is that of equalizing access to information. However, not much appears to be equal on the web. For example, the distribution of traffic and links to sites is extremely skewed and approximates a power law [5, 35], with a disproportionate share of traffic and links going to a small number of very popular sites. Evidence of a trend towards "winners take all" behavior can be seen in the market share of popular services. For example, the largest conventional book retailer (Barnes & Noble) has less than 30% market share, however the largest online book retailer (Amazon) has over 70% market share [52].

Search engines may contribute to such statistics. Prior to the web, consumers may have located a store amongst all stores listed in the phone book. Now, an increasing number of consumers locate stores via search engines. Imagine if most web searches for given keywords result in the same sites being ranked highly, perhaps with popularity measures incorporated into the selection and ranking criteria [43]. Even if only a small percentage of people use search engines to find stores, these people may then create links on the web to the stores, further enhancing any bias towards locating a given store. More generally, the experience of locating a given item on the web may be more of a common experience amongst everyone, when compared with previous means of locating items (for example, looking in the phone book, walking around the neighborhood, or asking a friend). Note that this is different to another trend that may be of concern – namely the trend towards less common experiences watching TV, for example, where increasing numbers of cable channels, and increasing use of the web, mean that fewer people watch the same programs.

Biases in access to information can be limited by using the appropriate search service for each query. While searches for stores on the major web search engines may return biased results, users may be able to find less biased listings in online Yellow Pages phone directories. As another example, when searching with the names of the U.S. presidential candidates in February 2000, there were significant differences between the major web search engines in the probability of the official candidate homepages being returned on the first page of results [60]. Similar searches at specialized political search engines may provide less biased results. However, the existence of less biased services does not prevent bias in information access if many people are using the major web search

engines. Searches at directory sites like Yahoo or the Open Directory may also be less biased, although there may be significant and unequal delays in listing sites, and many sites are not listed in these directories.

The extent of the effects of such biases depends on how often people use search engines to locate items, and on the kinds of search engines that they use. New search services that incorporate context, and further incorporation of context into existing search services, may increase competition, diversity, and functionality, and help mitigate any negative effects of biases in access to information on the web.

5 Summary

Search engines make an unprecedented amount of information quickly and easily accessible – their contribution to the web and society has been enormous. However, the "one size fits all" model of web search may limit diversity, competition, and functionality. Increased use of context in web search may help. As web search becomes a more important function within society, the need for even better search services is becoming increasingly important.

References

[1] E. Adar, D. Karger, and L. Stein. Haystack: Per-user information environments. In *Proceedings of the 1999 Conference on Information and Knowledge Management, CIKM*, 1999.

[2] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

[3] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. WebWatcher: A learning apprentice for the World Wide Web. 1995.

[4] Marko Balabanovic. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*, pages 378–385. ACM Press, New York, 1997.

[5] Albert-László Barabasi and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[6] Carol L. Barry. *The Identification of User Criteria of Relevance and Document Characteristics: Beyond the Topical Approach to Information Retrieval*. PhD thesis, Syracuse University, 1993.

[7] K. Bharat and M.R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[8] J. Boyan, D. Freitag, and T. Joachims. A machine learning architecture for optimizing web search engines. In *Proceedings of the AAAI Workshop on Internet-Based Information Systems*, 1996.

[9] S. Brin. Extracting patterns and relations from the World Wide Web. In *WebDB Workshop at EDBT 98*, 1998.

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World Wide Web Conference*, Brisbane, Australia, 1998.

[11] J. Budzik and K.J. Hammond. User interactions with everyday applications as context for just-in-time information access. In *Proceedings of the 2000 International Conference on Intelligent User Interfaces*, New Orleans, Louisiana, 2000. ACM Press.

[12] J. Budzik, K.J. Hammond, C. Marlow, and A. Scheinkman. Anticipating information needs: Everyday applications as interfaces to Internet information servers. In *Proceedings of the 1998 World Conference of the WWW, Internet and Intranet*, Orlando, Florida, 1998. AACE Press.

[13] Jay Budzik, Kristian J. Hammond, Larry Birnbaum, and Marko Krema. Beyond similarity. In *Proceedings of the 2000 Workshop on Artificial Intelligence and Web Search*. AAAI Press, 2000.

[14] O. Buyukkokten, J. Cho, H. García-Molina, L. Gravano, and N. Shivakumar. Exploiting geographical location information of web pages. In *Proc. of the ACM SIGMOD Workshop on the Web and Databases, WebDB*, 1999.

[15] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. Automatic resource list compilation by analyzing hyperlink structure and associated text. In *Proceedings of the 7th International World Wide Web Conference*, 1998.

[16] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: A new approach to topic-specific web resource discovery. In *8th World Wide Web Conference*, Toronto, May 1999.

[17] Junghoo Cho, Héctor García-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the Seventh World-Wide Web Conference*, 1998.

[18] Frans Coetzee, Eric Glover, Steve Lawrence, and C. Lee Giles. Feature selection in web applications using ROC inflections. In *Symposium on Applications and the Internet, SAINT*, San Diego, CA, January 8–12 2001.

[19] N. Craswell, P. Bailey, and D. Hawking. Server selection on the World Wide Web. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 37–46, 2000.

[20] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of Fifteenth National Conference on Artificial Intelligence, AAAI 98*, pages 509–516, 1998.

[21] B. D. Davison, A. Gerasoulis, K. Kleisouris, Y. Lu, H. Seo, W. Wang, and B. Wu. DiscoWeb: Applying link analysis to web search. In *Proceedings of the Eighth International World Wide Web Conference*, page 148, Toronto, Canada, 1999.

[22] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, 10–14 September 2000.

[23] Junyan Ding, Luis Gravano, and Narayanan Shivakumar. Computing geographical scopes of web resources. In *26th International Conference on Very Large Databases, VLDB 2000*, Cairo, Egypt, September 10–14 2000.

[24] D. Dreilinger and A. Howe. Experiences with selecting search engines using meta-search. *ACM Transactions on Information Systems*, 15(3):195–222, 1997.

[25] Gary Flake, Steve Lawrence, and C. Lee Giles. Efficient identification of web communities. In *Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 150–160, Boston, MA, August 20–23 2000.

[26] Susan Gauch, Guihun Wang, and Mario Gomez. ProFusion: Intelligent fusion from multiple, distributed search engines. *Journal of Universal Computer Science*, 2(9), 1996.

[27] D. Gibson, J. Kleinberg, and P. Raghavan. Inferring web communities from link topology. In *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, 1998.

[28] Eric Glover, Gary Flake, Steve Lawrence, William P. Birmingham, Andries Kruger, C. Lee Giles, and David Pennock. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet, SAINT*, San Diego, CA, January 8–12 2001.

[29] Eric Glover, Steve Lawrence, William Birmingham, and C. Lee Giles. Architecture of a metasearch engine that supports user information needs. In *Eighth International Conference on Information and Knowledge Management, CIKM 99*, pages 210–216, Kansas City, Missouri, November 1999.

[30] Eric J. Glover, Steve Lawrence, Michael D. Gordon, William P. Birmingham, and C. Lee Giles. Web search – your way. *Communications of the ACM*, 2000. Accepted for publication.

[31] Graphic, Visualization, and Usability Center. GVU's tenth WWW user survey (conducted October 1998), 1998.

[32] L. Gravano, C. Chang, H. García-Molina, and A. Paepcke. STARTS: Stanford proposal for Internet meta-searching. In *Proc. of the 1997 ACM SIGMOD International Conference on Management of Data*, pages 207–218, 1997.

[33] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: Text-source discovery over the Internet. *ACM Transactions on Database Systems*, 24(2), 1999.

[34] Luis Gravano and Héctor García-Molina. Generalizing GlOSS to vector-space databases and broker hierarchies. In *International Conference on Very Large Databases, VLDB*, pages 78–89, 1995.

[35] B.A. Huberman, P.L.T. Pirolli, J.E. Pitkow, and R.M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280:95–97, 1998.

[36] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, San Francisco, California, 25–27 January 1998.

[37] Andries Kruger, C. Lee Giles, Frans Coetzee, Eric Glover, Gary Flake, Steve Lawrence, and Cristian Omlin. DEADLINER: Building a new niche search engine. In *Ninth International Conference on Information and Knowledge Management, CIKM 2000*, Washington, DC, November 6–11 2000.

[38] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. In *AAAI-98 Workshop on AI and Information Integration*, 1998.

[39] N. Kushmerick, D. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *IJCAI 97*, pages 729–735, Nagoya, Japan, 1997.

[40] Steve Lawrence, Kurt Bollacker, and C. Lee Giles. Indexing and retrieval of scientific literature. In *Eighth International Conference on Information and Knowledge Management, CIKM 99*, pages 139–146, Kansas City, Missouri, November 1999.

[41] Steve Lawrence and C. Lee Giles. Context and page analysis for improved web search. *IEEE Internet Computing*, 2(4):38–46, 1998.

[42] Steve Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[43] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Nature*, 400(6740):107–109, 1999.

[44] Steve Lawrence and C. Lee Giles. Searching the web: General and scientific information access. *IEEE Communications*, 37(1):116–122, 1999.

[45] Steve Lawrence, C. Lee Giles, and Kurt Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[46] D. Leake, R. Scherle, J. Budzik, and K. Hammond. Selecting task-relevant sources for just-in-time retrieval. In *Proceedings of the AAAI-99 Workshop on Intelligent Information Systems*, Menlo Park, CA, 1999. AAAI Press.

[47] H. Lieberman. Letizia: An agent that assists web browsing. In *1995 International Joint Conference on Artificial Intelligence*, Montreal, CA, 1995.

[48] Media Metrix. Media Metrix announces top 25 digital media/web properties and sites for January 1999, 1999.

[49] W. Meng, K. Liu, C. Yu, W. Wu, and N. Rishe. Estimating the usefulness of search engines. In *15th International Conference on Data Engineering, ICDE*, Sydney, Australia, 1999.

[50] Stefano Mizzaro. Relevance: The whole history. *Journal of the American Society for Information Science*, 48(9):810–832, 1997.

[51] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & Webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence, AAAI*, 1996.

[52] Ivan Png. The competitiveness of on-line vis-a-vis conventional retailing: A preliminary study. In *11th NEC Research Symposium*, Stanford, CA, 2000.

[53] J. Rennie and A. McCallum. Using reinforcement learning to spider the web efficiently. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML-99)*, 1999.

[54] Bradley Rhodes. *Just-in-Time Information Retrieval*. PhD thesis, Massuchesetts Institute of Technology, 2000.

[55] Bradley J. Rhodes. Margin Notes: Building a contextually aware associative memory. In *Proceedings of the International Conference on Intelligent User Interfaces, IUI 00*, 2000.

[56] Bradley J. Rhodes and Thad Starner. Remembrance Agent: A continuously running automated information retrieval system. In *Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi Agent Technology*, pages 487–495, 1996.

[57] G. Salton. *Automatic text processing: the transformation, analysis and retrieval of information by computer*. Addison-Wesley, 1989.

[58] Kristie Seymore, Andrew McCallum, and Roni Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI 99 Workshop on Machine Learning for Information Extraction*, 1999.

[59] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272, 1999.

[60] D. Sullivan. Can you find your candidate? Search Engine Watch, February 29 2000.

[61] J. Xu and J. Callan. Effective retrieval with distributed collections. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 112–120, 1998.

[62] J. Zobel. Collection selection via lexicon inspection. In *Proceedings of the 1997 Australian Document Computing Symposium*, pages 74–80, Melbourne, Australia, 1997.

Searching for Needles in a World of Haystacks

Jamie Callan
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890, USA
callan+@cs.cmu.edu

Abstract

Current Web search engines are based on a single database *model of information retrieval. This paper argues that next generation Web search will be based on a* multi-database *model of information retrieval. The strengths, weaknesses, open research problems, and prospects of several multi-database retrieval models are discussed briefly.*

1 Introduction

Web search engines provide access to HTML documents that are stored on computers around the world. The distributed nature of the document collection provides the illusion of a complex information access task, but the first generation of Web search engines is actually based on a simple retrieval paradigm. Documents are copied from their original locations to a central site, where they are added to a central database, indexed, and made available for search. The first generation of Web search engines is based on a *single database* model of information retrieval.

The single database model of information retrieval was successful because initially i) the Web was not very large, ii) the Web was not a commercial medium, and iii) HTML pages were not very complex. A large, but not *too large*, amount of information was available for copying, freely, in a format that was easy to interpret. As the Web evolved, these conditions were violated. There was too much information for it all to be copied, access to some of it became restricted, and a large amount of information was in searchable databases or was generated dynamically in response to a request or user action. Web search engines today provide access to only a small portion of the information on the Web.

The single database approach to Web search is a poor foundation upon which to build next generation systems. New, multi-database architectures provide a more solid foundation, because they explicitly model the multi-site, multi-resource nature of the Web. Traditional Web search engines based on the single database model can still play a valuable role in a multi-database environment, but it will no longer be the central role.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

2 Multi-Database Retrieval Models

Multi-database retrieval models can be classified as appropriate for *small-scale* or *large-scale* environments. A *small-scale* environment might be one that contains a few hundred text databases, perhaps under the control of a single organization. A *large-scale* environment might be one that contains tens of thousands of text databases, perhaps under the control of multiple organizations.

We consider a multi-database retrieval model appropriate for a small-scale environment if human effort is required to maintain a database hierarchy, a knowledge base [13, 4], or training data [14] used for database selection. The determining factor is the ease with which a newly-discovered text database can be made accessible. We also consider a multi-database retrieval model appropriate for a small-scale environment if each query requires some degree of communication with every database [9].

Multi-database retrieval models for small-scale environments have their advantages. For example, manual database organization or the use of training data can enable very accurate database selection for typical queries. This approach to database selection is most likely to be used by commercial information services and in customer support centers, for example, where there is a premium on accuracy and less concern about the rapid introduction of new text databases on a regular basis.

We consider a multi-database retrieval model appropriate for a large-scale environment, such as the Web, if i) new text databases can be introduced quickly, easily, and automatically; and ii) only limited communication is required to decide which text databases a particular query should be sent to. Retrieval models that satisfy these requirements can be grouped into two classes: message-passing, and centralized selection.

3 Message-passing

A *message-passing* multi-database retrieval model is one in which a search request is repeatedly passed from one node in a computer network to adjacent nodes until either an answer is found or it reaches a search horizon. If each node passes requests to n neighbors, the search horizon at distance h contains n^h nodes, so search requests can be propagated rapidly to many computers in a distributed manner. Matching objects can be passed back along the search path (e.g., Freenet [5]), or a direct connection can be established between the original search node and the matching node(s) (e.g., Gnutella [10]).

Message-passing models are a truly distributed solution to searching across multiple databases. There is no central site that can be attacked or otherwise shut down [3]. Each node decides for itself whether it has content that matches the query, which makes it easy to integrate diverse information sources, dynamic information content, and resources that charge for content. It is also possible to offer a search client complete anonymity in a message-passing model [5], although doing so may entail significant communication costs for every node between the original search client and nodes with matching content.

Some characteristics of message-passing retrieval models are likely to limit their effectiveness as models for next-generation Web search systems. Perhaps the most serious is one of scale. If a message-passing model is applied on a large scale, the amount of additional Internet traffic and unnecessary computation is staggering. For example, a *single query* with fanout 4 and search horizon 10 would involve $\Sigma_{i=1}^{10} 4^i = 1,398,100$ nodes. It is difficult to imagine applying this model to millions of queries per day.

Even on a small-scale, message-passing models are *non-deterministic*, because answers depend upon the topology of the network at a particular time, and *non-optimal*, because only a portion of the network is searched for each query. Message-passing models are also prone to *spoofing*, in which a node claims to have matching information when it does not.

The limitations of the message-passing model are opportunities for interesting research. For example, issues of scale, non-determinism, and non-optimality might all be addressed by less random message-passing, while spoofing might be handled by holding each node responsible for its past actions. Whatever the solution, these

problems must be addressed before message passing architectures are a viable, large-scale solution to multi-database text retrieval.

4 Centralized Resource Selection

Perhaps the most fully developed multi-database retrieval models are based on centralized database selection. A central site gathers information about available databases, organizes it into an index, and then provides a resource selection service to search clients. A simple resource selection service might just return a ranked list of resources (searchable text databases) to the client. A more complex service might select a set of resources, submit the query to those resources, and return to the search client a single, merged list of documents obtained from multiple resources.

The best-known models for centralized database selection are based on the GlOSS family of algorithms [8], the CORI algorithm [1], and the Cue Validity Variance (CVV) algorithm [16]. The three algorithms differ in their details, but all are based on matching queries to "bag of words" representations of text databases. The "bag of words" resource representation is created automatically, making it relatively easy to add new resources to an existing resource index. The algorithms are general, relatively robust, and do not necessarily require the cooperation of the resource provider [1]. Unlike message-passing architectures, the centralized selection model is computationally efficient, is deterministic, and considers all resources for all queries.

Centralized services can be attacked, legally or illegally, and do not provide the anonymity that message-passing retrieval models provide. They also raise interesting research issues that are less likely to arise with message-passing retrieval models and retrieval models for small-scale environments.

Resource identification: Techniques for crawling the Web are well-defined, but it can be difficult to identify the type of resource at the other end of a hyperlink. For example, the Common Gateway Interface (CGI) hyperlink type indicates an executable resource, but the resource capabilities are not described. The resource might be a text database with search capabilities, a relational database, or a counter. Text associated with the link might or might not indicate something about its capabilities. A crawler that encounters a link to a CGI resource usually has few clues about the type of resource it points to.

Resource identification can be viewed as a document categorization task, for example assigning documents into categories such as "text database", "relational database", and "other" based on multiple forms of uncertain evidence.

Interoperability: Multi-database models designed for small-scale environments and message-passing retrieval models assume that all resources conform to a common interoperability standard. Multi-database retrieval models for large-scale environments are likely to find this assumption violated more often than satisfied. Rather than excluding such resources, it may be possible learn how to communicate with the resource via its Web page, for example, by probing it and examining its responses. Information can be extracted from returned Web pages using *wrapper induction* techniques [11]. Current wrapper induction algorithms are based on supervised learning, but large-scale interoperability requires wrapper induction algorithms based on unsupervised learning.

Resource representation: Content-based resource selection is based on knowing what subject area(s) each resource covers. For example, if the resource is an archive of Wall Street Journal newspaper articles, it might be described as covering the domains of stocks, bonds, international finance, politics, and similar subjects.

If the resource is a text database, and is controlled by a trusted and cooperative party, the STARTS protocol [7] is an effective method of discovering its contents. The STARTS protocol enables a search service and an information resource to exchange information about the contents and capabilities of the resource. If the

resource is controlled by an uncooperative party, the contents of the resource can be discovered by running queries and analyzing the documents that are returned (*query-based sampling*) [1].

Both STARTS and query-based sampling are sufficient for bag-of-words resource representations. More complex resource representations, for example, representations based on statistical language models [15], may require new approaches.

Resource selection: Given a set of information resources, content-based resource selection algorithms rank them by the likelihood that they will satisfy the information need described by a query. The GlOSS family of algorithms [8], CORI [1], and Cue Validity Variance (CVV) [16] are three examples that are based on variations of algorithms for ranking documents. The current generation of algorithms can be effective, but there is also evidence that they are fragile when expectations about query length, document length, and resource description quality are not met [2]. Even under ideal conditions, accuracy is considerably lower than theoretical limits [6].

Result merging: After a set information resources is searched, it may be desirable to merge the results into a single display or representation. This problem is difficult when resources are uncooperative. The state-of-the-art is either to rerank the documents at the search client, to merge the rankings using heuristics [1], or to reverse-engineer the ranking functions of the various search engines [12]. Each of these solutions has weaknesses, hence this remains an important, if often overlooked, research problem.

Robust, large-scale centralized resource selection is arguably the most complex of the multi-database retrieval models discussed above, because it requires solutions to several independent problems. Each of the problems outlined above is interesting and challenging because solutions must be general, automatic, and robust, and because they cannot rely on cooperation from information providers. The lack of cooperation, in particular, makes large-scale centralized resource selection both an appealing solution and an interesting research problem.

5 Forecast

All of the multi-database retrieval models surveyed in this paper are likely to play a role in next generation networked information systems. Models optimized for accuracy in small-scale environments will be used in corporate environments, and on the Web in specific subject areas (e.g., to select among databases covering medical research). Research efforts on this class of multi-database retrieval models will focus on improving accuracy and on reducing the amount of manual intervention necessary.

There are two adoption scenarios for message-passing retrieval models. Message-passing is a good choice for dynamic and diverse information content, because the information provider, which knows its content best, determines what information it can provide for each request. However, this same characteristic causes problems such as spoofing; centralized resource selection may provide adequate access and greater reliability. Message-passing is also a good choice when it is important to present no single point of information access, and when anonymity is important. The most obvious examples are theft of intellectual property such as music and software, and exchange of politically sensitive information.

Centralized resource selection is the retrieval model most likely to be the basis for the next generation of large-scale Web search systems. It scales well, it is effective, it provides consistent results, it supports heterogeneous text resources, and it supports heterogeneous query streams. There are research issues that must be addressed before multi-database retrieval models start becoming common, but they appear to be “near term” research opportunities, rather than “long term” research challenges.

Acknowledgements

This material is based on work supported in part by the National Science Foundation under grants IIS-9873009 and EIA-9983253. Any opinions, findings, conclusions or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor(s).

References

[1] J. Callan. Distributed information retrieval. In W.B. Croft, editor, *Advances in information retrieval*, chapter 5, pages 127–150. Kluwer Academic Publishers, 2000.

[2] J. Callan, A. L. Powell, J. C. French, and M. Connell. The effects of query-based sampling on automatic database selection algorithms. Technical Report IR-181, Center for Intelligent Information Retrieval, Department of Computer Science, University of Massachusetts, 1999.

[3] A. E. Cha. E-power to the people. In *The Washington Post*, page A01, May 18 2000.

[4] A. S. Chakravarthy and K. B. Haase. NetSerf: Using semantic knowledge to find Internet information archives. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4–11, Seattle, 1995. ACM.

[5] I. Clarke. Freenet - Front page. http://freenet.sourceforge.net/, August 2000.

[6] J. French, A. Powell, J. Callan, C. Viles, T. Emmitt, K. Prey, and Y. Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 238–245. ACM, 1999.

[7] L. Gravano, K. Chang, H. García-Molina, and A. Paepcke. STARTS Stanford proposal for Internet meta-searching. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data*, 1997.

[8] L. Gravano, H. García-Molina, and A. Tomasic. GlOSS: Text-Source Discovery over the Internet. *ACM Transactions on Database Systems*, 24(2):229–264, 1999.

[9] D. Hawking and P. Thistlewaite. Methods for information server selection. *ACM Transactions on Information Systems*, 17(1):40–76, 1999.

[10] Wego.com Incorporated. Welcome to gnutella. http://gnutella.wego.com/, August 2000.

[11] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, 1997.

[12] K. Liu, W. Meng, C. Yu, and N. Rishe. Discovery of similarity computations of search engines. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM)*. ACM, 2000.

[13] R. S. Marcus. An experimental comparison of the effectiveness of computers and humans as search intermediaries. *Journal of the American Society for Information Science*, 34:381–404, 1983.

[14] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 172–179, Seattle, 1995. ACM.

[15] J. Xu and W. B. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 254–261, Berkeley, 1999. ACM.

[16] B. Yuwono and D. L. Lee. Search and ranking algorithms for locating resources on the World Wide Web. In S. Y. W. Su, editor, *Proceedings of the 12th International Conference on Data Engineering*, pages 164–171, New Orleans, 1996.

Next Generation Web Search: Setting Our Sites

Marti A. Hearst
School of Information Management and Systems
University of California Berkeley
hearst@sims.berkeley.edu

Abstract

The current state of web search is most successful at directing users to appropriate web sites. Once at the site, the user has a choice of following hyperlinks or using site search, but the latter is notoriously problematic. One solution is to develop specialized search interfaces that explicitly support the types of tasks users perform using the information specific to the site. A new way to support task-based site search is to dynamically present appropriate metadata that organizes the search results and suggests what to look at next, as a personalized intermixing of search and hypertext.

1 Introduction

Surveys indicate that search engine user satisfaction has risen recently. According to one survey, 80% of search engine users say they find what they want all or most of the time [21]. This is a surprising result given average query length is still quite short – about 2 words. How can people be finding exactly what they want given such short queries?

Since no published large-scale analysis exists, we currently have to make guesses. I think the answer is that, as a gross generalization, most people use web search engines to find good starting points – home pages of web sites that discuss a topic of interest. A query on "horseradish" is very general; it does not indicate what it is the user wants to know about or do with horseradish, so the best thing a search engine can do is bring up sources of general information about horseradish which the user can then peruse in more detail.

This multi-stage process of search, starting with a general query and then getting more specific, is well-documented in non-web search [14]. Users of old search systems like Dialog and Lexis-Nexis were taught to first write a general query, look at how many (thousands) of results were returned, and then refine the query with additional terms until a reasonable number of documents resulted. In these older systems the user had to first select a collection, or source, to search. Many of these searchers were professionals, who knew a great deal about which sources were available after years of experience.

By contrast, in web search the purpose of the initial query seems primarily to be to choose the source – a web site of interest. Once at the source, the user has a choice of using hyperlinks to navigate through the many pages of information available, or using site search.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

In this article, I use the term "site" to mean a collection of information that has some kind of unified theme. This can be a collection of items such as architectural images, recipes, or biomedical texts, or the catalog of an e-commerce company, or the intranet of a company or a university (the theme being the various kinds of work done and information used by people in the organization), or what's begun to be called a "vortal" for vertical portal, which offers a wealth of different sources about one topic. FindLaw is an example of a vortal, providing search over dozens of different legal sources, including law journals, US Supreme court decisions, and legal news.

Major search engine companies are now offering site-specific search as part of their product suites. For example, Inktomi has announced plans to develop an architecture that allows flexible partitioning of information sets, allowing the assignment of weights to documents depending on which parts of the collection they occur in and what kinds of context they are associated with [19]. To best utilize this potentially powerful facility, an understanding is needed of how to combine this information effectively.

In the early days of web search, a query like "horseradish" would most likely have retrieved what felt like random pages. But two things have changed since then. First, more high-quality content has become available on a vast array of topics, and second, search engines now focus on returning definitive *sites*, rather than pages, for such general queries. For instance, the top hits for a search on "horseradish" on Lycos finds a link to the Horseradish Information Council, listed as part of the

> Recreation > Food
> Business > Industries > Food and Related Products > Fruit and Vegetables

web directory categories, and hits on encyclopedia entries for horseradish are shown alongside the

> Reference > Encyclopedia > Microsoft Encarta > H
> Reference > Encyclopedia > Encyclopedia.com > H

categories. These are followed by the home pages of various food products companies. (It is a pity that we cannot run this same query over the 1996 web using 1996 search engines for comparison purposes.)

The fact that search engines show hits on category labels is significant, because web directory categories, and their associated links, are manually selected to be representative starting points for search. The Google search engine is known for using hyperlink inlink information for ranking pages, based on the idea that if many pages link to a page, that linked-to page is likely to be of higher quality (recursively) because it is in effect "recommended" by the authors of the other pages. Others have also documented the merit of using inlink information for assessing page quality [12, 1], and other web search engines are making use of this information. However, Google also now incorporates category information into its search results, listing category labels beneath a search hit if that category had been assigned to it. An informal test on short general queries brought up on average 2.7 categories alongside the top 10 hits.[1]

A recent study [1] presents intriguing evidence that the *number of pages in the site* is a good predictor for the inlink-based ranking, at least for popular entertainment topics. In other words, the most popular sites on a topic according to inlink information are those sites that have a lot of information on the topic; the good sources. This bolsters the argument that what web search is good at doing is getting people to the right site or collection, after which the complicated information seeking begins.

2 Site Search and Tasks

In a well-designed web site, hyperlinks provide helpful hints about what is behind them and where to go next, a characteristic also known as "scent" [5, 16]. One usability expert claims that as a general rule, users do not object so much to following many links as they object to having to follow links that do not clearly support their task.

[1] The queries were horseradish, election, colon cancer, sex, berkeley, affirmative action, china, chat, recipes, united airlines.

If the user is unsure where to go next or has to resort to the back button, the usability of the site can decrease dramatically [20].

When the user has to resort to using the search facility, the results on a site are usually disorderly and shown out of context, and do not indicate what role the various hits play in the use or structure of the site. Usability gurus and ecommerce researchers alike lament the poor quality of site search, and claim that billions in business are lost each year due to poor site design and site search [20, 11].

What is a solution? Consider the following analogy. Following hyperlinks is like taking the train, whereas using site search is like driving an all-terrain four-wheel drive vehicle. On the train, there are a fixed number of choices of where to go and how to get there. To get from Topeka to Santa Fe you have to go through Frostbite Falls whether this make sense to you or not. On the other hand, you are unlikely to get lost – if you look at where you are on the map, all is clear. By contrast, a four-wheel drive Land Cruiser will take you anywhere, but you may get wedged between two boulders on the side of a cliff and be completely disoriented as to your whereabouts.

An ideal search interface adopts the best aspects of each technique. We would like a train system that magically lays down new track to suggest useful directions to go based on where we have been so far and what we are trying to do. The tracks follow the lay of the land, but cross over the crevasses and get to every useful part of the earth. They also allow us to back up at any time and take a different route at each choice point.

2.1 The Importance of the Task

Figuring out the best way to lay these tracks is nontrivial, because just as there is a huge variety of information available on these sites, there are also a huge variety of ways people use that information. To help cut down on the number of possible routes, the search interface, as well as the site structure, should reflect what is it people do with the site, that is, what *tasks* they attempt to accomplish on the site.

A recent study by Jared Spool's research firm uncovered the importance of task completion in user satisfaction in web site use [18]. The study compared 10 different sites; each participant conducted searches for information of interest to them on each site. Afterwards, Spool's team asked the participants to rate the web sites according to how fast they thought they were. Surprisingly, there was no correlation between the page download speeds and the perceived speed ratings. In fact, participants perceived the site with the fastest download speed to be the slowest, and vice versa.

However, there was a strong correlation between perceived speed and how successful the participants were at achieving their goals on the site. This was also correlated with how strongly the participant thought they usually "knew what to do next" at any given point in their task. Spool concludes that the correlational evidence suggests that if the goal is to improve perceived speed, it is more important to focus on designing web sites to help users complete their tasks, rather than focusing on task-neutral features like download speed.

I have now said that site search should reflect the tasks that a user would like to accomplish on the site, and I have suggested a metaphor about a magical train that lays tracks according to where you want to go and what you have done so far, a blending of the best features of hypertext and search. In the remainder of this section I will describe this idea in more detail and outline our research efforts in this direction.

2.2 Metadata

One more ingredient is needed before we can cook up this new idea. That is the notion of metadata. Metadata is commonly glossed as meaning "data about data". Most documents have some kinds of meta-information associated with them – that is, information that characterizes the external properties of the document, that help identify it and the circumstances surrounding its creation and use. These attributes include author(s), date of publication, length of document, publisher, and document genre.

Additionally, content-oriented subject or category metadata has become more prevalent in the last few years, and many people are interested in standards for describing content in various fields. Web directories such as

Yahoo and looksmart are familiar examples, and as seen above, web search engines have begun to interleave search hits on category labels with other search results.

Collections such as medical documents and architectural images have richer metadata available; some items have a dozen or more content attributes attached to them. It can be useful to think of category metadata as being composed of *facets*: orthogonal sets of categories, which together can be used to describe a topic. In the medical domain, the different facets are Disease type, Drug Type, Physiology, Surgery Type, Patient Type, and so on. Each article is a complex combination of several of these types of facets, each of which has a hierarchical structure. For example, a MedLine article entitled "Inhaled and systemic corticosteroid therapies: Do they contribute to inspiratory muscle weakness in asthma?" is assigned the MeSH categories *Steroidal Anti-Inflammatory Agents, Asthma, Muscular Diseases, Respiratory Muscles, Inhalation Administration, Adult, Beclomethasone, Case-Control Studies, Prednisone*, and *Risk Factors*, among others.

Researchers have long reported that using metadata in search is problematic, because the labels assigned often mismatch user expectations [15, 6]. Furthermore, category metadata is often inconsistent. Nevertheless, I believe the full potential of metadata in search results is underexplored and could yield significant improvements, especially for supporting task-based search over large collections of similar-style items (such as biomedical articles, architectural images, and recipes).

Metadata can be used as a counterpoint to free text, since free text and metadata can both be searched, but whereas free text queries must usually be subject to relevance ranking, metadata can be retrieved much as in a standard database query. It is much easier to accurately implement the query "Find all documents that have been assigned the category label Affirmative Action" than it is to implement "Find all documents about affirmative action".

2.3 An Example: epicurious

As a consequence of writing this paper, a website was brought to my attention that exhibits a good subset of the ideas I think can be useful for using metadata to improve task-oriented site search.[2] In this case the collection is recipe information; actually a database problem (fuzzy matching is not required) but many of the ideas can transfer to the fuzzier needs of information search.

Recipes are examples of information for which hierarchical faceted metadata is familiar to everyone. The facets used by epicurious are Main Ingredient, Cuisine, Preparation Method, Season/Occasion, and Course/Dish. Each of these has subcategories; for example, subcategories for Course/Dish include Appetizers, Bread, Desserts, Sandwiches, Sauces, Sides and Vegetables. The collection has over 11,000 recipes.

A standard search interface for recipes requires the user to either do a keyword search or drill down a category hierarchy. For example, on a different recipe site (called SOAR[3]), selecting Main Dishes $>$ Poultry results in 57 recipes. To further refine these choices, some additional hyperlinked categories are shown:

> Poultry: Chicken Recipes; Poultry: Duck Recipes; Poultry: Game Hens;
> Poultry: Game Recipes; Poultry: Goose; Poultry: Turkey

Selecting Chicken retrieves a list of about 40 recipes and two more subcategories:

> Chicken Appetizers
> Diabetic Chicken Recipes

I could have also found the Chicken Appetizers category if I'd begun with the Appetizers category. However, if I'd wanted to see Italian recipes with chicken as a main dish, I have to first select Region $>$ Italian, and then look

[2] http://www.epicurious.com/e_eating/e02_recipes/browse_main.html
[3] http://soar.berkeley.edu/recipes/

through all 665 Italian recipes. The site also allows a search for a particular term over the category, so I can do a search on "chicken" in the Italian section and hope for the best.

One problem with this interface is that I do not know how many recipes have a given attribute until after I follow the link. It would be much more useful to know this information when having to make a decision about where to go next. Researchers in the human-computer interaction community advocate the importance of information previews in this kind of situation [17]. Another flaw is the irregularity of what kinds of subcategories occur under a given category. Why are only appetizers shown as a special category, but not main dishes?

Recipe finding is in most cases a combination of a browsing and a search task, and so is a prime candidate for our ideas about combining the best of both techniques. The epicurious site does an excellent job of this. On this site, after selecting Main Ingredient > Poultry, the information of Figure 1 is shown.

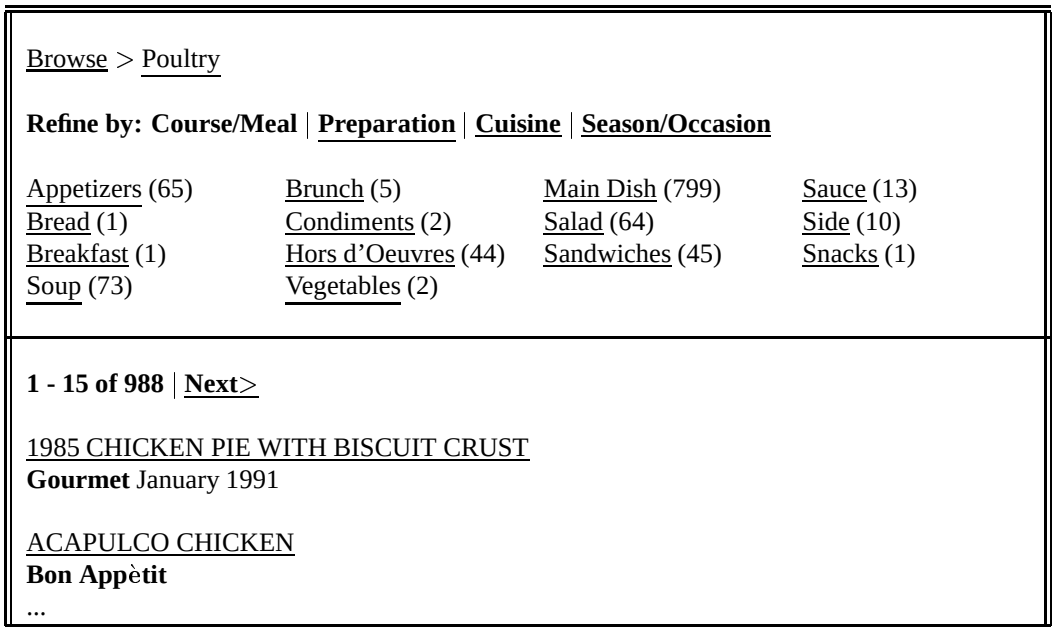

Figure 1: A sketch of the epicurious site's method for browsing dynamic metadata describing a collection of recipes. After selecting a main dish type (poultry), the user can refine the results using four other types of metadata.

This view allows me to reduce the set of recipes along any of the other metadata facets. It also tells me how many recipes will result if I refine the current metadata facet (Course/Meal) according to one if the terms in its subhierarchy. If I select Hors d'Oeuvres, the view of Figure 2 results.

Note that I have created this combination of categories on the fly. I could have started with Course Dish > Hors d'Oeuvres, and then refined by Main Ingredient > Poultry and ended up at the same point.

Now I can further refine the set of recipes by selecting a preparation type, the subhierarchies for which are shown above. Alternatively, I can select Cuisine and the system will show the same set of 44 chicken Hors d'Oeuvres recipes according to cuisine type, such as Caribbean, Italian, Low Fat, and Kid-Friendly.

The site also allows search over document subsets. In the example above, I can search over the 44 chicken appetizer recipes to isolate those that include avocado or some other ingredient. Search is also allowed over the entire dataset. However, the results of search are shown as a long unordered list of recipes, and thus loses the benefits of the browsing interface. There is, however, an advanced search form that allows the user to select sets of main ingredients along with the other metadata types. It suffers in comparison to the browsing facility, however, in not helping users avoid empty or large results sets.

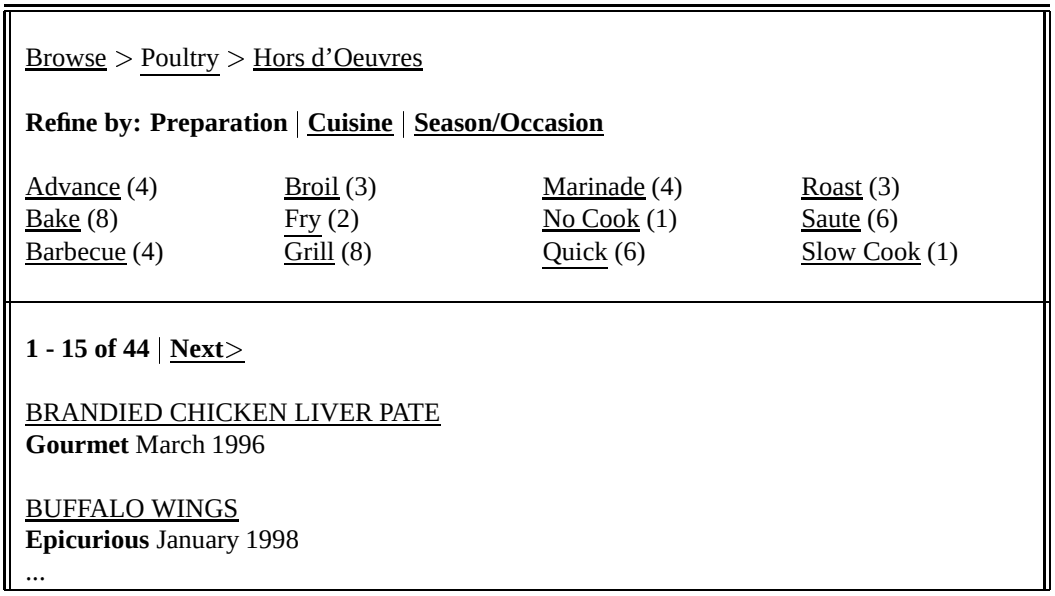

Figure 2: Result of revising the results of Figure 1 by selecting the Hors d'Oeuvres metadata type.

This interface supports information seeking for several different kinds of recipe-related tasks, including, e.g., "Help me find a summer pasta," (ingredient type with event type), "How can I use an avocado in a salad?" (ingredient type with dish type), and "How can I bake sea-bass" (preparation type and ingredient type). It does not support other tasks such as menu planning and organizing by customer reviews.

2.4 Example: Yahoo

The epicurious site provides a nice example of how to incorporate metadata into the search process, acting as a kind of dynamically-determined hyperlink. This is different from a setup like the directory structure at Yahoo, in which the paths are determined in advance. The Yahoo directory does combine certain types of metadata – most notably Region types are intermixed with other categories – but usually only up to two types are combined, and the combinations are not tailed to what the user wants to see. For example, to find UC Berkeley following links, the links that I clicked on were College and University > Colleges and Universities > United States > U > University of California > Campuses > Berkeley. This makes use of crosslinks (symbolic links), so the official category once I finally see a link for UC Berkeley is: U.S. States > California > Education > College and University > Public > University of California > Campuses. After clicking on Berkeley, the new category label that is actually associated with the information about UC Berkeley is U.S. States > California > Cities > Berkeley > Education > College and University > Public > UC Berkeley. This is an entirely different path than the one I traversed via hyperlinks. To handle the fact that most categories are best reached by multiple kinds of metadata, Yahoo does a great deal of crosslinking that causes the actual category labels traversed to change beneath the user. A system that lets the user choose which kinds of metadata to view next should be more effective.

By contrast to this clumsy use of metadata, Yahoo has a nice way of dynamically linking metadata in its restaurant selection site. At the top level it presents links that aid in tasks involving selection of restaurants, including maps and telephone directories. Before any searching can take place, the user must navigate a region metadata hierarchy to select a city. The user can then either select a cuisine link or issue a query over restaurant names or cuisine types, resulting in a list of restaurants that meet these criteria. After selecting a hyperlink for a particular restaurant, the user sees a summary of information about the restaurant which includes a set of links

grouped under the label of "Find Nearby". These links include movies, bars and clubs, and cafes that can be found in the geographic neighborhood of the selected restaurant. (Researchers are also providing this type of functionality [4].) There is also a link labeled "More A&E" (A&E indicates Arts and Entertainment, the supercategory for Restaurants and for Movies), but this breaks the conceptual model by showing a listing of all entertainment choices in the city, not those limited just to be near the selected restaurant.

In essence, Yahoo has assumed that many of those users searching the restaurant collection are actually engaged in a larger task which can be paraphrased as the stereotypical "find evening entertainment" task: dinner and a nearby movie. Two metadata facets are combined here: the region facet and the entertainment facet, and within this, two subhierarchies with the entertainment facet have been linked together – restaurant (with a cuisine attribute) and movies.

2.5 Integrating Search

The epicurious site does a nice job of dynamically suggesting metadata to help the user reduce the set of documents in an organized manner, making use of information previews to show how many documents would result after each choice, and allowing the user to easily back up to earlier states in the search process by clicking on the hyperlinks indicating the path taken so far. However, the interface does not interweave the search results into the category structure, and a straightforward improvement would be to organize the search results according to the same category metadata layout that is used for the browsing interface.

However, a large text collection such as the MedLine collection of biomedical abstracts is more difficult to search and organize than something like recipes. Additional facilities may be needed to apply this kind of dynamic metadata interface to something as complex and voluminous as medical text, and information retrieval-style ranking is probably necessary along with keyword search to help sort through results. Furthermore, the metadata is in some cases more hierarchical than in the recipe example, and more types of metadata are available, so only a subset should be shown at any given time. The system should dynamically determine which *types* of metadata to show, based on what the user has done so far and their past history (this idea has been pursued in other contexts [10, 13]). For example, a clinician prescribing medications for a patient may want to be able to always see categories associated with this patient's particular allergies.

2.6 Example: BioMedical Text

Consider the following example. Say a medical clinician named Dr. Care needs to find information about the use of cortisone shots as a treatment for asthma for a particular patient. Using our proposed system, Dr. Care can begin either by selecting an initial category label or by typing in some terms directly. Assume she already knows the MeSH category labels for the high-level concepts *Asthma* and *Steroids* but does not want to have to remember the category labels for more specific terms. By specifying these labels directly, the equivalent of a conjunctive Boolean search is run over the collection.[4] This returns 99 articles, which have a total of 2000 MeSH subject headings, 577 of which are unique.

Figure 3 shows an example of what an interface that incorporates the ideas discussed above might look like. The system provides Dr. Care with a way to get started in dealing with this large result set. It indicates the path taken to get to this point, the titles of two of the articles, suggestions for next steps below this, and the full document list at the bottom.

At the top is shown a hyperlinked path indicating the two choices made so far. The links allow the user to easily go back to an earlier stage in the navigation process; by selecting the Asthma link, Dr. Care would see the

[4]The data for this example was generated by running queries over the Medline/Healthstar database for 1995-1999 as of July, 1999, using the California Digital Library interface to this system. The collection contains article abstracts from 8,400 journals. Article information was downloaded and processed by hand in order to obtain the numbers. In some cases the details are simplified for expository purposes.

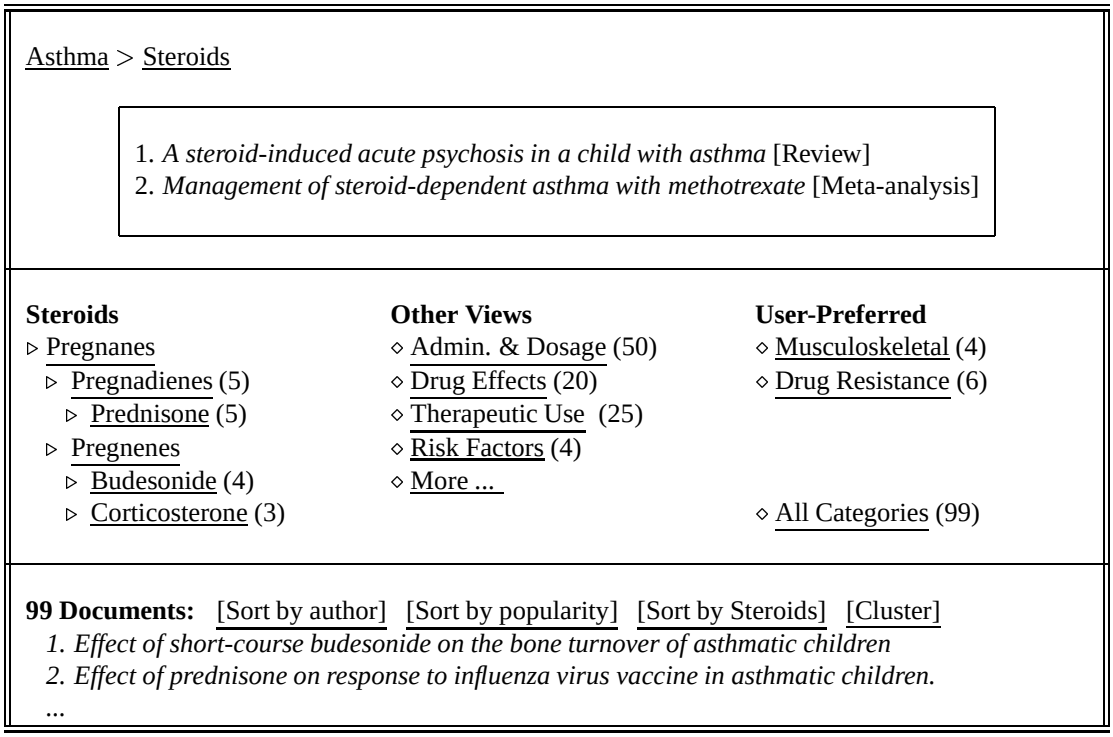

Figure 3: Sketch of a proposed method for browsing and searching biomedical text.

documents associated with this term, independent of Steroids.

Below this appears a box showing a few selected document titles. These are a review article and a meta-analysis, which are appropriate given the high-level terms used so far in the navigation process. Other reviews exist in this result set but talk about specific steroids and so are not shown at this point. This illustrates another important idea: a search interface should match the level of generality of the retrieved documents to the generality of the current navigation state. For example, at the early stages of the search, overview articles should be shown, but as the search becomes more specific, so should the documents.

Below the title box are shown lists of categories. The lefthand side shows a subset of the Steroids portion of the chemicals facet of MeSH. Only those categories within the Steroids subtree that occur significantly within the 99 documents of this result set are shown. The steroid metadata hierarchy implicitly shows which branch of the steroid family they occupy, along with the number of documents in this subcollection that refer to the particular steroid. Thus the user sees a preview of what would happen if they added any of these terms to the query; the results would be reduced dramatically. But rather than actually issuing a query, the user can simply click on the category label, thereby navigating to a portion of the search space that contains the steroid term conjoined with the other terms used so far. The user can subsequently easily back up to the current state by following the hyperlinked path at the top of the screen.

The righthand column shows how user-preferred categories can be integrated into the interface. Say Dr. Care is concerned about musculoskeletal and drug resistance issues for this particular patient. These categories will appear on all views of the results collection, along with a preview of how many documents are in the set. These preferences could be specified directly by the user, inferred from previous selections, or based on citation structure or popularity in terms of page accesses. Finally, the user can elect to see all categories that correspond to documents in the current set, if they prefer to override the system's organization facilities.

The bottommost portion of the display allows Dr. Care to scan the retrieval results directly, and also allows

her to reorder the titles in various ways. Sorting by popularity takes into account how often other users have viewed the documents. It also provides a "Sort by Steroids" option. This appears because Steroids is the most recently chosen category (which is also why it appears in the lefthand category column). The resulting ordering would make use of the structure of the Steroids subhierarchy to group documents with similar steroids together. Finally, users can invoke a clustering option to show a textual or graphical display of documents according to their overall commonality [9].

As these examples from disparate fields show, there are commonalities in supporting effective search strategies between domains. In our new research project, FLAMENCO, we are investigating these ideas of FLexible information Access using MEtadata in Novel COmbinations. Our end goal is to develop a general methodology for specifying task-oriented search interfaces across a wide variety of domains and tasks. We suggest that rich, faceted metadata be used in a flexible manner to give users information about where to go next, and to have these suggestions and hints reflect the users' individual tasks.

3 Other Approaches to Site Search

3.1 Specialized Interfaces

Another way to improve web site search is to create a specialized interface that takes the structure of the information on the site into account. One of our research projects applies a variation of this idea in an attempt to improve Intranet search. Intranets contain the information associated with the internal workings of an organization, and our system, called Cha-Cha, organizes web search results in such a way as to reflect the underlying structure of the organization. An "outline" or "table of contents" is created by first recording the shortest paths in hyperlinks from root pages to every page within the web intranet. After the user issues a query, these shortest paths are dynamically combined to form a hierarchical outline of the context in which the search results occur [3]. For example, hits on the query "earthquake" will be shown to fall within the mechanical engineering department, a science education project, and administrative pages that indicate emergency evacuation plans.

This interface has been deployed as the UC Berkeley site search engine for the last two years, receiving on average about 200,000 queries a month. Based on user interviews and surveys, it is sometimes quite useful to see the context in which the search hit occurred, especially when the query does not produce a good hit. On the other hand, the extra information can be overwhelming and unnecessary if the search is relatively straightforward. Furthermore, an Intranet's structure does not always reflect the user's task; often some other kind of organization would be more appropriate. For example, a user trying to find out about research on the effects of old-growth logging probably cares about the different kinds of logging under consideration, but not which university department the results came from.

Another example of a search interface that follows the structure of the information is the CiteSeer interface [7] which focuses on a *type* of information – scientific references – rather than a content domain. The search structure reflects the structure of the underlying information: citations have hyperlinks to other papers by the same authors, and to paragraphs of text of articles in which the target article is cited. Search results of aggregates of users are exploited both for ranking and for producing informative statistics, such as how many articles cite a particular author's papers. The interface shows special links that would not make sense in general search, and are tailored to what people searching research literature are interested in.

3.2 Question Answering

I believe a number of other approaches to web search will flourish as time goes on. These include rent-an-expert sites, where users are matched up with people who have expertise in a field and answer their questions for a fee.

Some services connect clients and experts via the phone, avoiding the necessity of typing and having the potential to spread easily to use via mobile devices (Keen.com and Exp.com are two examples).

Systems to automate question answering are also improving, and these will become important supplements for organizations' web sites, to handle customer questions more quickly and cheaply. Rather than generating an answer from scratch, these systems attempt to link a natural language query to the most pertinent sentence, paragraph, or page of information that has already been written. They differ from standard search engines in that they make use of the structure of the question and of the text from which the answers are drawn. For example, a question about what to do when a disk is full needs to be linked to an answer about compressing files or buying new disk. A user asking "Why does my computer keep hanging?" wants to find information about how to avoid this situation, whereas a user asking "How do I get my computer to print?" wants information about how to bring the situation about. The syntax of the question, as well as the content words, determines the kind of acceptable answer.

Question answering in a limited domain can be very powerful but it is much harder in a broad domain. Sophisticated question answering has not yet gotten far in web search aside from the well-known example of AskJeeves, in which question types are manually linked in advance to specific answer pages. However, researchers [2, 8] and companies (such as AnswerLogic and InQuizit) are developing domain-specific natural language processing algorithms and lexical resources that should greatly improve automated question answering in the next two to three years.

Technologists are becoming highly interested in what might be considered "real-world" metadata. The thinking is that a query of "Where is a good Mazda mechanic?" should automatically take note of the local time and location of the question asker, in order to make suggestions of car repair places that are both nearby and open at the time the question is asked. The need for such context-aware questions answering systems can be expected to grow along with the demand for networked mobile devices.

4 Integration with General Web Search

Returning to the original topic of this essay, what is the role of general search engines in the framework proposed above? General search engines should evolve to direct people to task-oriented solutions, instead of collection-oriented solutions as the modus operandi today. In other words, search engines will need to match user requests to task descriptions. One step in this direction is to interpret multi-word queries in terms of their implicit task. For example, to use document genre to determine which results to return. As a straightforward example, a query on "review" alongside another term such as the name of a play should bring back sites with theatre reviews. Currently the unstated default for most queries, for ad servers at least, is that the user's task is to buy something. Eventually I envision task directories supplementing directories, and search engines providing search over these descriptions.

Acknowledgements
I would like to thank Jan Pedersen for many helpful conversations about web search, Doug Cook for detailed information about Inktomi's current search results statistics, and Eric Brewer, Ame Elliott, Dan Glaser, Luis Gravano, Jason Hong, Rashmi Sinha, and Hal Varian for helpful comments on this article.
This research was supported an NSF CAREER grant, NSF9984741.

References

[1] Brian Amento, Loren Terveen, and Will Hill. Does 'authority' mean quality? Predicting expert quality ratings on web documents. In *Proceedings of the 23rd Annual International ACM/SIGIR Conference*, pages 296–303, Athens, Greece, 2000.

[2] Claire Cardie, Vincent Ng, David Pierce, and Chris Buckley. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, pages 180–187. Association for Computational Linguistics/Morgan Kaufmann, May 2000.

[3] Michael Chen, Marti A. Hearst, Jason Hong, and James Lin. Cha-cha: A system for organizing intranet search results. In *Proceedings of the 2nd USENIX Symposium on Internet Technologies and Systems*, Boulder, CO, October 11-14 1999.

[4] Junyan Ding, Luis Gravano, and Narayanan Shivakumar. Computing geographical scopes of web resources. In *Proceedings of the Twenty-sixth International Conference on Very Large Databases (VLDB'00)*, Sept 2000.

[5] George W. Furnas. Effective view navigation. In *Proceedings of ACM CHI 97 Conference on Human Factors in Computing Systems*, volume 1 of *PAPERS: Information Structures*, pages 367–374, 1997.

[6] Fredric Gey, Hui-Min Chen, Barbara Norgard, Michael Buckland, Youngin Kim, Aitao Chen, Byron Lam, Jacek Purat, and Ray Larson. Advanced search technologies for unfamiliar metadata. In *Meta-Data '99 Third IEEE Meta-Data Conference*, Bethesda, MD, April 1999.

[7] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 1998.

[8] Sanda Harabagiu, Marius Pasca, and Steven Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the COLING-2000*. Association for Computational Linguistics/Morgan Kaufmann, Aug 2000.

[9] Marti A. Hearst, David Karger, and Jan O. Pedersen. Scatter/gather as a tool for the navigation of retrieval results. In Robin Burke, editor, *Working Notes of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, Cambridge, MA, November 1995. AAAI.

[10] Haym Hirsh, Chumki Basu, and Brian D. Davison. Learning to personalize. *Communications of the ACM*, 43(8), Aug 2000.

[11] Mark Hurst. Holiday '99 e-commerce. http://www.creativegood.com, Sept 1999.

[12] Jon Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[13] Henry Lieberman. Letizia: an agent that assists web browsing. In *Proceedings of 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 924–929, 1995.

[14] Gary Marchionini. *Information Seeking in Electronic Environments*. Cambridge University Press, 1995.

[15] Karen Markey, Pauline Atherton, and Claudia Newton. An analysis of controlled vocabulary and free text search statements in online searches. *Online Review*, 4:225–236, 1982.

[16] Peter Pirolli. Computational models of information scent-following in a very large browsable text collection. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 3–10, Vancouver, Canada, May 1997. ACM.

[17] Catherine Plaisant, Ben Shneiderman, Khoa Doan, and Tom Bruns. Interface and data architecture for query preview in networked information systems. *ACM Transactions on Information Systems*, 17(3):320–341, 1999.

[18] Tara Scanlon, Will Schroeder, Richard Danca, Nina Gilmore, Matthew Klee, Lori Landesman, Amy Maurer, Paul Sawyer, and Jared Spool. *Designing Information-Rich Web Sites*. User Interface Engineering, 1999.

[19] Chris Sherman. Inktomi inside. http://websearch.about.com/internet/websearch/library/weekly/aa041900a.htm, April 2000.

[20] Jared Spool. *Web Site Usability: A Designer's Guide*. Morgan Kaufmann, 1998.

[21] Danny Sullivan. NPD search and portal site study. http://searchenginewatch.internet.com/reports/npd.html, July 6 2000. NPD's URL is http://www.npd.com.

ICDE 2001

in the Heart of Europe

The 17th International Conference on Data Engineering

April 2–6, 2001
Heidelberg, Germany

The 17th International Conference on Data Engineering (ICDE 2001) will be held in a beautiful old town – Heidelberg. This is at the center of Germany's most active high-tech region where you find world renowned companies, numerous software startups, many world-class research centers such as the ABB Research Center, the Deutsche Telekom Research and Development Center, the European Media Laboratory, the European Molecular Biology Laboratory, the Fraunhofer Gesellschaft Institute for Graphical Data Processing, the German Center for Cancer Research, the GMD National Center for Information Technology, the Computer Science Research Center, the Karlsruhe Research Center, and a number of universities with strong database research groups (amongst others Darmstadt University of Technology, International University in Germany, University of Karlsruhe, University of Mannheim and not far away the University of Stuttgart). Such combination of strong industry, ground breaking research institutions, economic prosperity, and a beautiful host town provide an ideal environment for a conference on Data Engineering.

With tutorials, panels and industrial program.

Website & Registration at
http://www.congress-online.de/ICDE2001

Early Registration until the 2nd of March 2001

Topics Include:

XML, METADATA, and SEMISTRUCTURED DATA
DATABASE ENGINES & ENGINEERING
QUERY PROCESSING
DATA WAREHOUSES, DATA MINING, AND KNOWLEDGE DISCOVERY
ADVANCED IS MIDDLEWARE
SCIENTIFIC AND ENGINEERING DATABASES
EXTREME DATABASES
E-COMMERCE and E-SERVICES
WORKFLOW and PROCESS-ORIENTED SYSTEMS
EMERGING TRENDS
SYSTEM APPLICATIONS AND EXPERIENCE

Conference Officers

General Chairs:	
Andreas Reuter	European Media Laboratory and International University in Germany
David Lomet	Microsoft Research, USA
Program Co-chairs:	
Alex Buchmann	University of Darmstadt, Germany
Dimitrios Georgakopoulos	Telcordia Technologies, USA
Panel Program Chair:	
Erich Neuhold	GMD-IPSI, Germany
Tutorial Program Chair:	
Guido Moerkotte	University of Mannheim, Germany
Eric Simon	INRIA, France
Industrial Program Co-chairs:	
Peter Lockemann	University of Karlsruhe, Germany
Tamer Ozsu	University of Alberta, Canada
Steering committee liaison:	
Erich Neuhold	GMD-IPSI, Germany
Marek Rusinkiewicz	MCC, USA
Organizing Chair:	
Isabel Rojas	European Media Laboratory, Germany
Demos & Exhibits:	
Wolfgang Becker	International University in Germany
Andreas Eberhart	
Local Arrangements:	
Bärbel Mack	European Media Laboratory, Germany
Claudia Spahn	
Public Relations:	
Peter Saueressig	European Media Laboratory, Germany

IEEE COMPUTER SOCIETY

Sponsored by the IEEE Computer Society

Financial Support: Special support for travel expenses and conference fees will be available for participants from Eastern Europe.

IEEE Computer Society
1730 Massachusetts Ave, NW
Washington, D.C. 20036-1903

Non-profit Org.
U.S. Postage
PAID
Silver Spring, MD
Permit 1398

A Heuristic-Based Methodology for Semantic Augmentation of User Queries on the Web*

Andrew Burton-Jones[1], Veda C. Storey[1], Vijayan Sugumaran[2], and Sandeep Purao[3]

[1] J. Mack Robinson College of Business, Georgia State University,
Atlanta, GA 30302,
{vstorey,abjones}@gsu.edu

[2] School of Business Administration, Oakland University
Rochester, MI 48309
sugumara@oakland.edu

[3] School of Information Sciences & Technology, The Pennsylvania State University,
University Park, PA 16801-3857
spurao@ist.psu.edu

Abstract. As the World Wide Web continues to grow, so does the need for effective approaches to processing users' queries that retrieve the most relevant information. Most search engines provide the user with many web pages, but at varying levels of relevancy. The Semantic Web has been proposed to retrieve and use more semantic information from the web. However, the capture and processing of semantic information is a difficult task because of the well-known problems that machines have with processing semantics. This research proposes a heuristic-based methodology for building context aware web queries. The methodology expands a user's query to identify possible word senses and then makes the query more relevant by restricting it using relevant information from the WordNet lexicon and the DARPA DAML library of domain ontologies. The methodology is implemented in a prototype. Initial testing of the prototype and comparison to results obtained from Google show that this heuristic based approach to processing queries can provide more relevant results to users, especially when query terms are ambiguous and/or when the methodology's heuristics are invoked.

1 Introduction

It is increasingly difficult to retrieve relevant web pages for queries from the World Wide Web due to its rapid growth and lack of structure [31, 32]. In response, the Semantic Web has been proposed to extend the WWW by giving information well-defined meaning [3, 10]. The Semantic Web relies heavily on ontologies to provide taxonomies of domain specific terms and inference rules that serve as surrogates for semantics [3]. Berners-Lee et al. describe the Semantic Web as "not a new web but an extension of the current one, in which information is given well-defined meaning" [3]. Unfortunately, it is difficult to capture and represent meaning in machine-

* This research was partially supported by J. Mack Robinson College of Business, Georgia State University and Office of Research & Graduate Study, Oakland University.

I.-Y. Song et al. (Eds.): ER 2003, LNCS 2813, pp. 476–489, 2003.

readable form, even though understanding more of the semantics of a user's application or query would help process users' queries more effectively.

There is wide agreement that a critical mass of ontologies is needed for representing semantics on the Semantic Web [6, 27]. Libraries of ontologies are being developed for this purpose, with the most well-known being the DARPA DAML library with approximately 200 ontologies and over 25,000 classes http://www.daml.org/ontologies/. Although significant effort has gone into building these ontologies, there is little research on methodologies for retrieving information using them. Thus, the development of a methodology for doing so would greatly assist in realizing the full potential of the Semantic Web.

The objective of this research, therefore, is to: *develop a heuristic-based methodology for an intelligent agent to process queries on the Semantic Web so that the processing takes into account the semantics of the user's request*. This is done through the use of WordNet [15] (http://www.cogsci.princeton.edu/cgi-bin/webwn) to obtain senses for query terms and WordNet and the DAML ontologies to augment a query by expanding and shrinking the set of query terms to achieve a more precise, context-specific query.

The contribution of this research is to develop a methodology that more effectively processes queries by capturing and augmenting the semantics of a user's query. The methodology has been implemented in a prototype and its effectiveness verified. Results of this research should help realize the potential of the Semantic Web, while demonstrating useful applications of lexicons and ontology libraries.

2 Related Work

2.1 Semantic Web

The unstructured nature of the web makes it difficult to query and difficult for applications to use. The Semantic Web is a vision of the web in which these problems will be solved by 'marking-up' terms on web pages with links to online ontologies that provide a machine-readable definition of the terms and their relationship with other terms [3]. This is intended to make the web machine-readable and, thus, easier to process for both humans and their applications (e.g., agents). Consider a prototypical query (adapted from [3]): *Find Mom a specialist who can provide a series of bi-weekly physical therapy sessions*. To complete this query on the Semantic Web, agents will use online ontologies to interpret relevant semantics on web pages (e.g., specialist, series, etc) [16, 25]. The proliferation of ontologies is crucial for the Semantic Web, hence the creation of large ontology libraries [10, 47]. Hendler [25] predicts: "*The Semantic Web…will not primarily consist of neat ontologies...I envision a complex Web of semantics ruled by the same sort of anarchy that rules the rest of the Web*." Therefore, research is needed to determine how useful these ontologies will be for helping users and agents query the Semantic Web.

2.2 Ontologies

An ontology should be a way of describing one's world [50]. Ontologies generally consist of terms, their definitions, and axioms relating them [19]. However, there are many different definitions, descriptions, and types of ontologies [20, 39, 40]. In knowledge representation, well known contributions include Ontolingua [14], SHOE [23], Cyc [21] and the XML based schemes such as OIL [16], and DAML [24]. Ontologies can be characterized as either formal/top-level ontologies that describe the world in general or material/domain ontologies that describe specific domains [20]. Semantic Web ontologies (e.g., at the DAML library) are primarily domain ontologies. The development of domain ontologies was motivated by the need to develop systems that could reason with common sense knowledge of the real world [19]. More formally, a domain ontology is a catalog of the types of things that are assumed to exist in the domain of interest, D, from the perspective of a certain language, L, for the purpose of talking about that domain, D [45]. Conceptual modeling researchers have contributed extensively to the development and application of both formal and domain ontologies [2, 11, 13, 30, 50].

2.3 Information Retrieval

Methods from the information retrieval (IR) field [42] can inform the process of querying the Semantic Web and the role of domain ontologies. A core problem in IR is word-sense disambiguation: a word may have multiple meanings (homonymy), yet several words can have the same meaning (synonymy) [26, 36]. Resolving homonymy increases the relevance of the results returned (precision) by eliminating results of the wrong word-sense; resolving synonymy increases the proportion of relevant results in the collection returned (recall) by including terms that have the same meaning.

In IR, word-sense disambiguation involves two steps: 1. identifying the user's intended meaning of query terms, and 2. altering the query so that it achieves high precision and recall. In IR, the first step is usually achieved by automatically deducing a term's meaning from other terms in the query [1]. This is feasible because IR queries are typically long, e.g., 15 terms for short queries [9] and 50-85 for long queries [22]. On the Semantic Web, however, this appears infeasible. Most web queries are only two words long [46] and this is an insufficient length to identify context [9, 49]. Therefore, some user interaction will be required to accurately identify the intended sense of query-terms [1].

The second step (altering the query) is generally achieved in IR through a combination of:

- query constraints, such as requiring pages to include all query terms (possibly near each other) [22, 37].
- query expansion with 'local context,' in which additional terms are added to the query based on a subset of documents that the user identifies as relevant [37, 43]
- query expansion with 'global context,' in which additional terms are added to the query from thesauri, from terms in the document collection, or from past queries [9, 18, 29, 41, 49]

Of these methods, query constraints should be useful on the Semantic Web because they improve web search [38]. Query expansion with local context is less likely to be effective because reports indicate that web users rarely provide relevance feedback [8, 46]. Finally, query expansion with global context will remain relevant, but the sources used in global context analysis will be largely superseded by ontologies. For example, rather than use a thesaurus to add synonyms to a query so that relevant pages were not missed, a Semantic Web query could be left unexpanded and could simply rely on web pages referencing the terms on their pages to ontologies that defined each term and its synonyms. Recall, however, that ontologies will be of mixed quality [25]. Therefore, thesauri will likely remain important on the Semantic Web. The IR field suggests that two thesauri should be used (ideally in combination) [29, 34, 38, 49]: (1) general lexical thesauri that detail lexically related terms (e.g., synonyms), and (2) domain-specific thesauri that detail related terms in a specific domain. The preferred lexical thesaurus is WordNet, a comprehensive on-line catalog of English terms [15]. WordNet classifies the English language into synonym sets with underlying word senses (e.g., the noun "chair" has 4 word senses). WordNet has been found useful for traditional and web IR [38, 49]. It is difficult, however, to identify domain-specific thesauri for all domains on the web [18]. A solution is to use the domain ontology libraries on the Semantic Web. Stephens and Huhns [47] have shown that large ontology libraries can provide useful knowledge even in the presence of individual ontologies that are incomplete or inaccurate. Ontology libraries, therefore, have a dual role on the Semantic Web: 1) as a source of definitions of terms on specific web pages, and 2) as a source of semantics that can assist query expansion.

In summary, the IR field provides several insights into methods that will be required for querying the Semantic Web. The following insights, in particular, have influenced the development of the methodology presented in this paper:

- the need for user-interaction to identify the context of terms in short web queries,
- the continued relevance of query expansion using global context analysis,
- the need to use lexical and domain thesauri in combination, and
- the important role of large libraries of domain ontologies as sources of semantics.

3 Methodology for Retrieving Information from Semantic Web

Semantic Web languages and query schemes are still being developed [7, 10]. Nonetheless, the usefulness of Semantic Web ontologies for querying can be tested on the current web. Consider Berners-Lee's et al. query: *Find Mom a specialist who can provide a series of bi-weekly physical therapy sessions*. The need for disambiguation is clear if the query is transformed into a short, more ambiguous query, more closely approximating queries on the web [46]: *Find doctors providing physical therapy*.

3.1 Overview of Methodology

A methodology for processing the query above is presented in Figure 1, above. Steps 1 to 3 identify the query context, as outlined below.

Identify Query Context	→	Build Knowledge Base	→	Execute Query
Step 1:		Step 4:		Step 7:
Identify noun phrases		Exclude incorrect sense		Construct Boolean query
Step 2:		Step 5:		Step 8:
Build synonym sets		Include hypernyms		Run query on search engine
Step 3:		Step 6:		Step 9:
Select word sense		Resolve inconsistencies		Retrieve and present results

Fig. 1. Methodology for Disambiguating, Augmenting, and Executing Query

Step 1 – Identify noun phrases. The methodology assumes that users enter their queries in natural language form, as users often have difficulty using Boolean logic or other query syntax [33, 46]. Nouns are identified using a modified form of the Qtag part-of-speech tagger [35]. Identifying phrases can significantly improve query precision [9, 33]. Thus, noun phrases are identified by querying each consecutive word-pair in WordNet. For example, "physical therapy" is a phrase in WordNet, so this step would find two noun phrases ('doctor' and 'physical therapy'). These noun phrases form the base nodes for expansion of the query. They are represented by a semantic network (see Figure 2). Initially, the terms are lined by a 'candidate' relationship, indicating that the terms have not yet been identified as relating in a lexical or domain-specific way. In the following stages, the semantic network will be augmented with three additional relationships: synonym (X is the same as Y), hypernym (X is a subclass of Y), and negation (X is not Y).

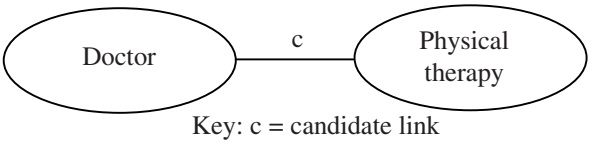

Fig. 2. Semantic Network of Query Terms after Step 1

Step 2 – Build synonym sets. To identify the different senses for each noun-phrase, synonym sets for each context are obtained from WordNet. For example, the term "doctor" has four senses in WordNet: 1. a medical practitioner, 2. a theologian, 3. a game played by children, and 4. an academician. "Physical therapy" has just one sense in WordNet. Each synonym set comprises one to many synonyms. These synonym sets for the query terms are incorporated into the knowledge base.

Step 3 – Select word sense. As the query does not contain enough terms to automatically deduce the word sense, user interaction is required [1, 49]. The user is presented with synsets for terms with multiple senses (e.g., "doctor") from which the user selects the most appropriate sense. Once the word-sense of terms has been identified, steps 4-6 build the knowledge base with additional terms to expand and constrain the query. The emphasis is on building a query that is biased towards precision, i.e., gives greater weighting to precision than recall [9, 22].

Step 4 – Exclude incorrect sense. To ensure that the results returned for a query are accurate, it is important to filter out those pages that contain incorrect senses of the term. Traditional query expansion does not include negative terms as filters [12, 18, 49]. Studies of web-query expansion have similarly not considered incorrect senses [28, 38]. Excluding incorrect senses is important on the web because of the vast number of results returned. Given that WordNet can return multiple synsets for each query term, incorrect senses can be inferred from the user's chosen word sense. For example, if a user selects the "medical practitioner" sense of 'Doctor,' then, a correct inference is that the user does not want pages associated with the other three senses of the term (theologians, children's games, or academics). Furthermore, because WordNet orders its synsets by estimated frequency of usage, terms can be chosen that are most likely to be successful in eliminating irrelevant results. We therefore use the following exclusion heuristic: *For each noun phrase, if a user identifies a synset as relevant, select the synonyms from the highest-ordered remaining synset, and include them in the knowledge base as negative knowledge* (see, e.g., Figure 3).

Step 5 – Including hypernyms. Past research has added hypernyms (or superclasses) to queries as a recall-enhancing technique, retrieving pages that contain either the search term (e.g., physical therapy) or its superclass (e.g., therapy) [18, 49]. Because precision is preferred over recall on the web [9, 22], our methodology uses hypernyms to increase the precision of queries by including them as mandatory terms. For example, rather than searching for *doctor OR "medical practitioner*," pages would be searched that contain *doctor AND "medical practitioner*." Of course, some 'doctor' pages that are relevant to the user may not include the hypernym "medical practitioner." Nevertheless, pages that contain 'doctor' and 'medical practitioner' are expected to be more likely to be consistent with the medical sense of the term 'doctor' than if the page only contained the 'doctor' term alone. Recall is, thus, sacrificed for precision. Following this approach, hypernyms are obtained from both WordNet and the DAML ontology library. WordNet provides the hypernyms for each synset automatically. The hypernyms from the DAML ontology library are obtained by querying the library for each noun phrase. The hypernyms from DAML and WordNet are then incorporated into the knowledge base. Table 1 shows the terms extracted from WordNet and the DAML ontology library from this step based upon the example. Figure 3 illustrates the expanded semantic network of terms after completing this step.

Step 6 – Resolve inconsistencies. The ability of query expansion to improve a query is dependent upon the quality of the terms added. Inconsistent terms could be added to the knowledge base when: (a) the synonym sets in WordNet are not orthogonal so a word-sense may be partially relevant but excluded by our methodology in step 4, (b) the DAML ontologies are of mixed quality so might contain inaccurate information [25], and (c) WordNet and the domain ontologies represent the contribution of many individuals who may have conflicting views of a domain. To identify inconsistencies, the methodology uses the following heuristic: *Check the hypernyms of the query terms (from DAML and WordNet) against the synonyms of the query term (from WordNet) that the user did not select as the desired word sense. Upon finding a match, ask the user if the term is desired. Adjust the knowledge base accordingly*.

Once the knowledge-base has been expanded, steps 7–9 are used to build and execute the query.

Table 1. Extraction of Hypernyms from WordNet and the DAML Ontology Library

Knowledge from WordNet		
Term	Word-sense (defined by synsets)	Hypernym (superclass)
Doctor	*Doc, Physician, MD, Dr, medico*	*Medical practitioner, medical man*
	Doctor of Church	*Theologian, Roman Catholic*
	Doctor	*Play*
	Dr	*Scholar, scholarly person, student*
Physical Therapy	*Physiotherapy, physiatrics*	*Therapy*
Knowledge from DAML ontology library		
Term	Hypernym (superclass)	
Doctor	Qualification, Medical care professional, Health professional	
Physical therapy	Rehabilitation, Medical practice	

Step 7 – Construct Boolean query. Following [22, 38], we construct the query using Boolean constraints to improve precision. Three heuristics are used:

(a) Synonym: *Automatically include the first synonym in the synset selected by the user and include it in the query with an OR, e.g., (query term OR synonym).*

(b) Hypernym: *Automatically include the first hypernym from WordNet for the user's selected word-sense. Allow the user to select up to one hypernym from those from the DAML library. Require the results to include either one or both hypernyms, e.g., query term AND (WordNet hypernym OR DAML hypernym).*

(c) Negation: *Automatically include the first synonym from the first synset in WordNet's list that was not selected by the user with a Boolean NOT, e.g., (query term NOT synonym).*

The rationale for each heuristic is to improve precision while minimizing user interaction. The synonym heuristic is used because the user may not have selected the most precise query term for that word sense. While this heuristic on its own could increase recall at the cost of precision [18], the use of a synonym in combination with a hypernym should improve precision. Because WordNet lists terms in estimated order of frequency of use, the first synonym is likely the best alternative for that term.

The hypernym heuristic is used to force pages to include the query term (or synonym) as well as the hypernym should increase the likelihood that the page contains the sense of the term the user desires. Following [34], WordNet and the DAML ontology are used in combination by allowing either of their hypernyms to be found. Because the DAML ontologies can be of mixed quality and because they do not provide hypernyms by word sense, user interaction is required to select the appropriate hypernym obtained from the domain ontologies.

Finally, the negation heuristic is used to filter out unwanted word senses. As WordNet can provide many synonyms in each synset and because terms are listed in estimated order of frequent usage, the first synonym from the first remaining synset is chosen as the most useful term for excluding pages.

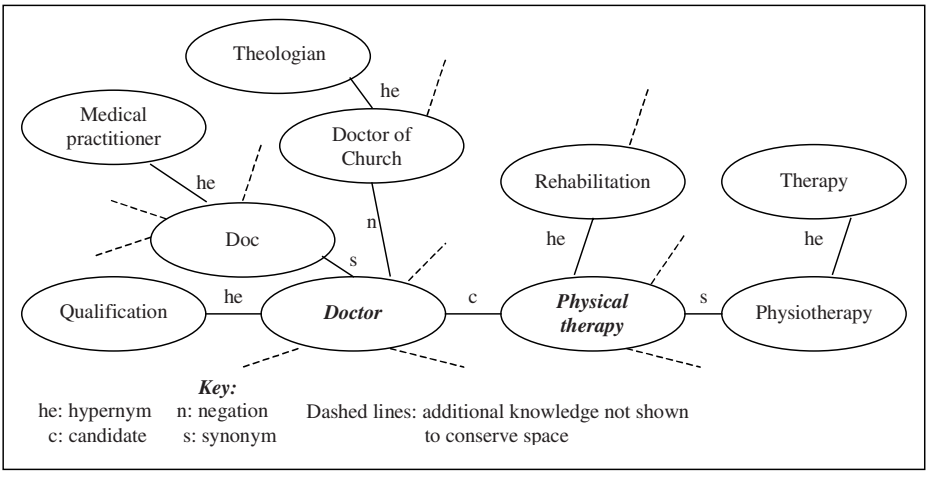

Fig. 3. Semantic Network of Query-Terms After Building Knowledge-Base

Applying these heuristics to the example, the following query is constructed: *(Doctor or Doc) and ("Medical practitioner" or Qualification) –"Doctor of church" and ("Physical therapy" or physiotherapy) and (therapy or "medical practice").*

Step 8 – Run query on search engine. Although a number of IR search engines have been developed, (e.g., for TREC), there is evidence that web search engines are more effective for web querying [44]. The methodology, therefore, submits the query to one or more web search engines (in their required syntax) for processing. The query construction heuristics are designed to work with most search engines. For example, while Altavista.com allows queries to use a NEAR constraint, this is unavailable on other search engines (e.g., Google.com, Alltheweb.com), so it is not used. Likewise, query expansion methodologies in IR can add up to 800 terms to the query with varying weights [41]. This approach is not used in our methodology because web search engines limit the number of query terms (e.g. Google has a limit of ten terms).

Step 9 – Retrieve and present results. In the final step, the results from the search engine (URLs and 'snippets' provided from the web pages) are retrieved and presented to the user. The user can either accept the query or revise the query to get more relevant results.

A pilot-test of the methodology indicated that the approach is feasible [5]. Queries using the above steps returned fewer results of equal or more relevance, with results dependent on the length and ambiguity of the query. A prototype was thus implemented to enable more detailed testing.

4 Implementation

A prototype called ISRA (Intelligent Semantic web Retrieval Agent) has been developed using J2EE technologies and informally tested [48]. ISRA uses the traditional client-server architecture as shown in Figure 4. The client is a basic web browser, through which the user specifies search queries in natural language. The server contains Java application code and the WordNet database. The prototype also provides an interface to several search engines including Google (www.google.com), Alltheweb (www.alltheweb.com) and AltaVista (www.altavista.com).

The prototype consists of three agents: a) Input-Output-Parser Agent, b) WordNet Agent, and c) Query Refinement and Execution Agent. These were implemented using a combination of jsp pages and servlets. The input-output-parser agent is responsible for capturing the user's input, parsing the natural language query, and returning results. The agent uses "QTAG", a *probabilistic parts-of-speech tagger* (available at http://web.bham.ac.uk/o.mason/software/tagger/index.html), to parse the user's input. It returns the part-of-speech for each word in the text. Based on the noun phrases (propositions) identified, an initial search query is created.

The WordNet Agent interfaces with the WordNet lexical database via JWordNet (a pure Java standalone object-oriented interface available at http://sourceforge.net/projects/jwn/). The prototype uses WordNet 1.6 (PC). For each noun phrase, the agent queries the database for different word senses and requests that the user select the most appropriate sense for the query. The agent extracts word senses, synonyms and hypernyms (superclasses) from the lexical database and forwards them to the query refinement agent to augment the initial query.

The Query Refinement and Execution (QRE) agent expands the initial query based on word senses, and synonyms obtained from WordNet. The refined query is then submitted to the search engine using appropriate syntax and constraints, and the results returned to the user. For example, the agent interacts with Google through its Web API service and adheres to the ten word limit for query length and displays ten hits at a time. Essentially, the QRE agent applies the steps shown in Figure 1 to augment the initial query. For example, it searches for phrases (word pairs) and includes them within double quotes, adds synonyms (from the WordNet synset) to the query based on the word sense selected by the user, and adds negative knowledge (terms) from the remaining word senses (e.g., would exclude the theologian sense of doctor if a query was for a medical doctor). For each term, the hypernym corresponding to the selected word sense is also retrieved from WordNet and DAML ontology and added to the query. The refined query is then sent to the search engine.

5 Testing

The effectiveness of the methodology was assessed by carrying out a laboratory study in which the results obtained using ISRA were compared to those obtained using the Google search engine alone. As the control group (Google) and the experimental group (Google plus methodology) used the same search engine, the experiment directly tests the benefit of the methodology and its heuristics.

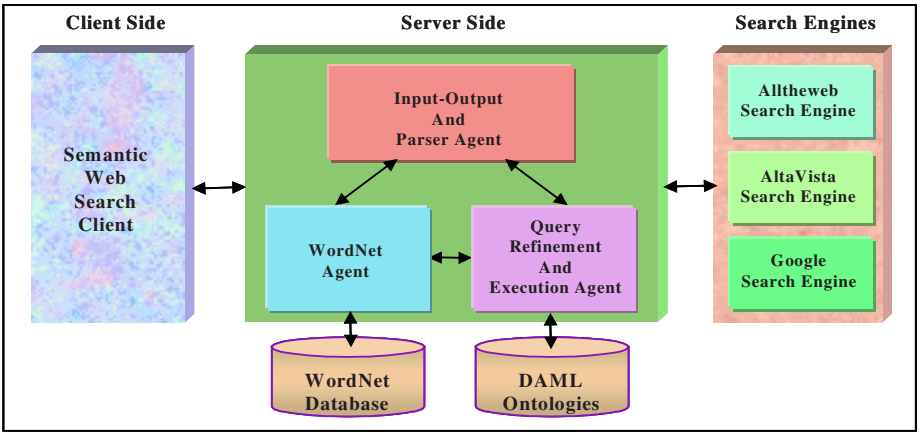

Fig. 4. ISRA Prototype Design

Forty-nine students from two universities participated voluntarily. All were experienced and frequent search engine users. Subjects were required to build their own queries and evaluate their own results since experimenters are unable to create as diverse a set of queries as users, can bias results, and cannot objectively determine whether a result is relevant to a user [17].

5.1 Dependent Variable and Hypothesis

Two measures of precision were used: the number of relevant pages in the first 10 and first 20 pages (Precision(10) and Precision(20)). These suit a web context because users do not view many results [4, 46]. Recall was not tested because it is less relevant and not strictly measurable on the web [12].

To test the flexibility of the methodology, the experiment tested the system's performance on two types of terms: ambiguous and clear. *Query term ambiguity* refers to the degree to which the query terms contain many word senses. Because the methodology aims to improve query precision, it should provide the most benefits when query terms have many word senses. Formally, the following hypothesis was tested:

Hypothesis: *ISRA will produce more relevant results than Google for queries with ambiguous terms.*

5.2 Query Sample

A large body of diverse user-developed queries was tested (per [4]). Some queries were expected to contain terms that did not exist in WordNet or the DAML library, e.g., 'instance' information such as product names. To alleviate this problem, and to test the hypothesis, the query sample was constructed as follows. First, all single-word classes in the DAML library were extracted. These were then pruned by excluding terms that (a) had no superclasses in the library, or (b) would be unknown to subjects

(e.g., very specialized terms). 268 terms were obtained. These were divided into two groups based on their word-senses in WordNet. 131 terms (the clear group) had 0-2 word-senses (where 0 represents a domain-specific term not in WordNet). 137 terms (the ambiguous group) had 3 or more word-senses. In the experiment, subjects constructed four queries. Two queries could use any words. Two were constrained to a random list of 20 terms from DAML. Subjects were randomly assigned either 20 clear or unclear terms. Subjects used these to construct the two constrained queries.

5.3 Procedure

Each subject received instructions explaining the system's operation, how to construct queries, and how to grade web-page relevance. Subjects were first given a 10-minute training exercise to introduce them to the system and to practice developing a query and ranking results. Next, each participant developed his or her four queries. Each student's materials included the random list of clear or unclear terms for constructing their two constrained queries. The system provided the results from both ISRA and Google (in random order) and asked the subject for his or her ranking. Subjects were given three minutes to perform each query and four minutes to rank the first twenty pages returned for each query (two minutes to rank the results from ISRA and two minutes to rank the results from Google). A short time was given for assessment because this was considered to be typical web user behavior. Overall, the experiment took approximately 45 minutes.

5.4 Summary of Results

Table 2 summarizes the results. For each query, the methodology was expected to increase the number of relevant pages, decrease the number of irrelevant pages, and reduce the number of hits. Paired-sample t-tests identify the difference between groups. To assess the benefit of the heuristics, all subjects' queries were analyzed to determine if they invoked the query construction heuristics in section 3.1 (step 7) above. A more detailed analysis of each heuristic (e.g., to compare the contribution of terms from WordNet versus terms from DAML) is being examined in future research. The results for the experiment are:

All differences between the system and Google were in the expected direction

- For all queries, the methodology significantly reduced the number of pages returned
- For the full sample: the number of irrelevant pages in the top 10 was significantly reduced
- For the ambiguous queries: all results were significant and supported the hypothesis
- For the full sample, results were strong when heuristics were invoked. Using synonyms, hypernyms, and/or negative knowledge decreased the irrelevant pages. Using synonyms and/or negative knowledge increased the relevant pages.

Table 2. Summary of Results

Variable	Hypothesized Direction	All	Unclear	Syn	Hyp	Neg	SNH
R(10)	+	+.279	+.054*	+.091*	+.175	+.033**	+.003**
R(20)	+	+.371	+.014**	+.180	+.453	+.050*	+.006**
NR(10)	-	-.035**	-.003**	-.002**	-.007**	-.006**	-.000**
NR(20)	-	-.106	-.002**	-.011**	-.034**	-.007**	-.000**
Hits	-	-.000**	-.000**	-.000**	-.000**	-.000**	-.000**
N	NA	156	44	96	123	95	63

Key: Cell entries are p-values for paired sample t-tests (ISRA vs. Google).
** significant at $\alpha < 0.05$ one-tailed, * at $\alpha < 0.10$ one-tailed
R(10) = # relevant in top 10 , R(20) = # relevant in top 20, NR = # not relevant (in top 10 & 20), N = Sample size, All = full set of queries minus missing values, Unclear = subset of queries for the ambiguous group, Syn = subset of queries that invoked synonym heuristic, Hyp = subset of queries that invoked hypernym heuristic, Neg = subset of queries that invoked negation heuristic, SNH = subsets of queries that invoked synonym, hypernyms, and negation heuristics.

6 Conclusion

A significant body of research has emerged to investigate ways to facilitate the development of the Semantic Web. There have been few attempts, however, to develop methodologies for retrieving information from the Semantic Web. This paper presented a methodology for the development of an intelligent agent that makes effective use of lexicons and ontologies for processing queries on the Semantic Web. The methodology is based upon a desire to obtain good results from a query with minimal user intervention. The methodology builds upon research on natural language processing, knowledge-based systems, and information retrieval. Initial results show that the approach is beneficial. A prototype system was found to improve query results over a common search engine when (a) query terms were ambiguous, and/or (b) when the methodology's heuristics were invoked. Further work is needed to improve the scalability and customizability of the approach, and minimize user interaction.

References

1. Allan, J. and H. Raghavan. *Using Part-of-Speech Patterns to Reduce Query Ambiguity*. in *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2002. Tampere, Finland: ACM Press, New York.
2. Bergholtz, M. and P. Johannesson. *Classifying the Semantics in Conceptual Modelling by Categorization of Roles*. NLDB'01. 2001. Madrid, Spain.
3. Berners-Lee, T., J. Hendler, and O. Lassila, *The Semantic Web*, in *Scientific American*. 2001. p. 1–19.
4. Buckley, C. and E.M. Voorhees. *Evaluating Evaluation Measure Stability*. in *SIGIR*. 2000. Athens, Greece: ACM.

5. Burton-Jones, A., S. Purao, and V.C. Storey. *Context-Aware Query Processing on the Semantic Web*. in *Proceedings of the 23rd International Conference on Information Systems*. 2002. Barcelona, Spain, Dec. 16–19.
6. CACM, *Special issue on ontology.* Communications of ACM, Feb. 2002. 45(2) p. 39–65.
7. Crow, L. and N. Shadbolt, *Extracting Focused Knowledge From the Semantic Web.* International Journal of Human-Computer Studies, 2001. 54: p. 155–184.
8. Cui, H., et al. *Probabilistic Query Expansion Using Query Logs*. in *Eleventh World Wide Web Conference (WWW 2002)*. 2002. Honolulu, Hawaii.
9. de Lima, E.F. and J.O. Pedersen. *Phrase Recognition and Expansion for Short, Precision-biased Queries based on a Query Log*. in *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1999. Berkeley, CA.
10. Ding, Y., et al., *The Semantic Web: Yet Another Hip?* Data & Knowledge Engineering, 2002. 41: p. 205–227.
11. Dullea, J. and I.Y. Song. *A Taxonomy of Recursive Relationships and Their Structural Validity in ER Modeling*. Lecture Notes in Computer Science 1728. 1999. Paris, France.
12. Efthimiadis, E.N., *Interactive Query Expansion: A User-Based Evaluation in a Relevance Feedback Environment.* Jl. of American Society for Inf. Science, 2000. 51(11) p. 989–1003.
13. Embley, D.W., et al. *A Conceptual-Modeling Approach to Extracting Data from the Web*. in *17th International Conference on Conceptual Modeling, ER '98*. 1998. Singapore.
14. Farquhar, A., R. Fikes, and J. Rice. *The Ontolingua Server: a Tool for Collaborative Ontology Construction*. in *Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop*. 1996. Banff, Canada.
15. Fellbaum, C. *WordNet: An Electronic Lexical Database*. 1998, MIT Press Cambridge, MA.
16. Fensel, D., et al., *OIL: An Ontology Infrastructure for the Semantic Web.* IEEE Intelligent Systems, 2001. March/April: p. 38–45.
17. Gordon, M. and P. Pathak, *Finding Information on the World Wide Web: The Retrieval Effectiveness of Search Engines.* Info. Processing & Management, 1999. 35: p. 141–180.
18. Greenberg, J., *Automatic Query Expansion via Lexical-Semantic Relationships.* Journal of the American Society for Information Science, 2001. 52(5): p. 402–415.
19. Gruber, T.R., *A Translation Approach to Portable Ontology Specifications.* Knowledge Acquisition, 1993. 5: p. 199–220.
20. Guarino, N. *Formal Ontology and Information Systems*. in *1st International Conference on Formal Ontology in Information Systems*. 1998. Trento, Italy: IOS Press.
21. Guha, R.V. and D.B. Lenat, *Enabling Agents to Work Together.* Communications of the ACM, 1994. 37(7): p. 127–142.
22. Hearst, M.A. *Improving Full-Text Precision on Short Queries using Simple Constraints*. in *SDAIR*. 1996. Las Vegas, NV.
23. Heflin, J., J. Hendler, and S. Luke, *SHOE: A Knowledge Representation Language for Internet Applications. Technical Report CS-TR-4078 (UMIACS TR-99-71)*. 1999, Dept. of Computer Science, University of Maryland at College Park.
24. Hendler, J. and D.L. McGuinness, *The DARPA Agent Markup Language.* IEEE Intelligent Systems, 2000. 15(6): p. 67–73.
25. Hendler, J., *Agents and the Semantic Web.* IEEE Intelligent Systems, 2001. Mar p. 30–36.
26. Ide, N. and J. Veronis, *Introduction to the Special Issue on Word Sense Disambiguation: The State of the Art.* Computational Linguistics, 1998. 24(1): p. 1–40.
27. IEEE, *Special issue on the Semantic Web.* IEEE Intelligent Systems, Mar 2001: p. 32–79.
28. Jansen, B.J., *An Investigation Into the Use of Simple Queries on Web IR Systems.* Information Research: An Electronic Journal, 2000. 6(1): p. 1–13.
29. Jing, Y. and W.B. Croft. *An Association Thesaurus for Information Retrieval*. in *RIAO-94, 4th International Conference ''Recherche d'Information Assistee par Ordinateur''*. 1994.
30. Kedad, Z. and E. Metais. *Dealing with Semantic Heterogeneity During Data Integration. 18th Intl Conf on Conceptual Modeling, LNCS 1728*. 1999. Paris, France.

31. Kobayashi, M. and K. Takeda, *Information Retrieval on the Web.* ACM Computing Surveys, 2000. 32(2): p. 144–173.
32. Lawrence, S., *Context in Web Search.* IEEE Data Engg. Bulletin, 2000. 23(3) p. 25–32.
33. Lewis, D.D. and K. Spark Jones, *Natural Language Processing for Information Retrieval.* Communications of the ACM, 1996. 39(1): p. 92–101.
34. Mandala, R., T. Tokunaga, and H. Tanaka. *Combining Multiple Evidence from Different Types of Thesaurus for Query Expansion*. in *22nd Intl ACM SIGIR Conference on Research & Development in Information Retrieval*. 1999. Berkeley, CA.
35. Mason, O., *Qtag – a Portable POS Tagger*. 2003, online: http://web.bham.ac.uk/O.Mason/software/tagger/.
36. Miller, G.A., *Contextuality*, in *Mental Models in Cognitive Science*, J. Oakhill and A. Garnham, Editors. 1996, Psychology Press: East Sussex, UK. p. 1–18.
37. Mitra, M., A. Singhal, and C. Buckley. *Improving Automated Query Expansion*. in *21st Intl Conf on Research & Development on Information Retrieval*. 1998. Melbourne, Australia.
38. Moldovan, D.L. and R. Mihalcea, *Improving the Search on the Internet by using WordNet and Lexical Operators.* IEEE Internet Computing, 2000. 4(1): p. 34–43.
39. Mylopoulos, J., *Information Modeling in the Time of the Revolution.* Information Systems, 1998. 23(34): p. 127–155.
40. Noy, N.F. and C.D. Hafner, *The State of the Art in Ontology Design: A Survey and Comparative Review.* AI Magazine, 1997. 18(3/Fall): p. 53–74.
41. Qiu, Y. and H.-P. Frei. *Concept Based Query Expansion*. in *16th Annual Intl ACM SIGIR Conference on Research and Development in Information Retrieval*. 1993. Pittsburgh, PA.
42. Raghavan, P. *Information Retrieval Algorithms: A Survey*, *Eighth Annual ACM-SIAM Symp on Discrete Algorithms*. 1997. New Orleans, Louisiana: ACM Press, New York, NY.
43. Salton, G. and C. Buckley, *Improving Retrieval Performance by Relevance Feedback.* Journal of the American Society for Information Science, 1990. 41(4): p. 288–297.
44. Singhal, A. and M. Kaszkiel. *A Case Study in Web Search Using TREC Algorithms*. in *10th International World Wide Web Conference*. 2001. Hong Kong.
45. Sowa, J.F., *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. 2000: Brooks Cole Publishing Co.
46. Spink, A., et al., *Searching the Web: The Public and Their Queries.* Journal of the American Society for Information Science, 2001. 52(3): p. 226–234.
47. Stephens, L.M. and M.N. Huhns, *Consensus Ontologies: Reconciling the Semantics of Web Pages and Agents.* IEEE Internet Computing, 2001. September–October: p. 92–95.
48. Sugumaran, V., A. Burton-Jones, and V.C. Storey. *A Multi-Agent Prototype for Intelligent Query Processing on the Semantic Web*. in *Proceedings of the 12th Annual Workshop on Information Technology and Systems (WITS)*. 2002. Barcelona, Spain, Dec. 14–15.
49. Voorhees, E.M. *Query Expansion Using Lexical-Semantic Relations*. in *17th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*. 1994. Dublin, Ireland.
50. Weber, R., *Ontological Foundations of Information Systems*. Coopers and Lybrand Accounting Research Methodology, Monograph No. 4. 1997, Melbourne: Coopers & Lybrand and Accounting Association of Australia and New Zealand. 212.

Retrieving Passages and Finding Answers

Mostafa Keikha, Jae Hyun Park, W. Bruce Croft, Mark Sanderson[1]
CIIR, University of Massachusetts Amherst, Amherst, MA
[1]RMIT University, Melbourne, Australia
{keikham, jhpark, croft } @cs.umass.edu, mark.sanderson@rmit.edu.au

ABSTRACT

Retrieving topically-relevant text passages in documents has been studied many times, but finding non-factoid, multiple sentence answers to web queries is a different task that is becoming increasingly important for applications such as mobile search. As the first stage of developing retrieval models for "answer passages", we describe the process of creating a test collection of questions and multiple-sentence answers based on the TREC GOV2 queries and documents. This annotation shows that most of the description-length TREC queries do in fact have passage-level answers. We then examine the effectiveness of current passage retrieval models in terms of finding passages that contain answers. We show that the existing methods are not effective for this task, and also observe that the relative performance of these methods in retrieving answers does not correspond to their performance in retrieving relevant documents.

1. INTRODUCTION

Different queries can be answered with different granularities of text. For example, "factoid" queries can be answered with single facts or named entities and navigational queries can be answered with a web page. However, "informational" queries, especially longer queries that take the form of a question, can have answers as small as a sentence or as big as multiple web pages. Our hypothesis is that there are many queries for which a passage-level, multiple-sentence answer can be superior to a document-level answer and, for those queries, result lists that include passages will be more effective than documents alone. Many of the "description" queries from TREC topics, such as "What allegations have been made about Enron's culpability in the California energy crisis?" (topic 737), are examples of the type of query for which passage-level answers should be appropriate. These queries, however, cannot be as simply categorized as the queries in factoid QA systems, nor can relatively simple templates be used to identify answers.

ADCS '14 November 27 - 28 2014, Melbourne, VIC, Australia

http://dx.doi.org/10.1145/2682862.2682877.

An area where effective passage-level answer retrieval could have broader impact is mobile search applications with limited output bandwidth based on using either a small screen or speech output. In this case, the ability to use "answer passages" to reduce the amount of output while maintaining high relevance will be critical. The popularity of recent voice search applications shows the enormous potential for an effective speech-based mobile search tool. Answer passages could also be used instead of snippets in the result page, which would help users to identify answers more quickly.

Although it is potentially an important task, answer passage retrieval has not been studied in the context of non-factoid web queries. As the first stage of a comprehensive study of this task, we build a data set based on the TREC GOV2 collection that contains passage-level answers to some of the description queries. We then explore the effectiveness of existing passage retrieval models for the task of finding answers. The main two questions we address in this paper are whether passage-level answers exist for longer web queries, and whether these passage-level answers can be found using existing passage retrieval methods.

2. RELATED WORK

Passage retrieval has been studied in the information retrieval community from different perspectives. Using passage-level information to improve document ranking has been the most common approach [3, 2]. Passages have also been used for query expansion [8] and as the first stage of factoid question answering systems [13].

The effectiveness of passage retrieval models has primarily been judged by the effectiveness of the document ranking that is generated using the passages. Less attention has been paid to directly retrieving passages as final answers to a query. This problem was partly addressed in the HARD track in TREC and the INEX ad hoc track where one of the tasks was to retrieve passages instead of documents [1, 6]. However, in these studies, the goal was to retrieve topically relevant parts of the document. A text passage that is topically relevant, however, does not necessarily provide an answer to the query. The closest effort to our study was the TREC Genomics track where the goal was to answer biomedical questions using a collection of scientific articles [5]. This study, however, was limited to the biomedical domain and there has been no similar study over open domain web queries and web collections. In this paper, we describe our effort to create a collection for answer retrieval using web documents. Further, we describe the results of current passage retrieval methods for the task of finding answers.

3. ANNOTATING ANSWER PASSAGES

Our experiments are based on the TREC GOV2 collection and its corresponding description-length queries. We hired two undergraduate students who were asked to highlight answer passages using an annotation system and a graduate student who performed quality control.

We divide topics randomly into two different groups, one for each annotator. For each topic, we retrieve the top 50 documents using the Sequential Dependence Model (SDM), a state-of-the-art retrieval model [9]. From the retrieved documents, we select the relevant documents (based on the TREC judgments) for the passage annotation phase. Each assessor annotates all the documents related to their assigned topics. They also annotated the top five documents for the other topics to study the agreement between annotators.

Guidelines were established to improve consistency. An answer passage is defined as a piece of text in a document that can answer a user information need defined in the description part of a TREC topic. Passages were evaluated based on whether they were complete (contains an entire answer) and concise (contains no irrelevant information).

Since typically there are only a few complete and concise passages with respect to each query, we relaxed the criteria somewhat and defined four levels of relevance (Perfect, Excellent, Good, Fair) based on the annotator's view of the quality of the answer passage. The definition for "excellent", as an example, was that only simple inference based on the user's background knowledge was required to answer the question using the passage, and that most of the passage is related to the answer.

Our annotators found 8,027 answer passages to 82 TREC queries, which is about 97 passages per query on average. Among all the annotated passages, 43% of them are perfect answers, 44% are excellent, 10% are good and the rest are fair answers. The annotators highlighted a total of 84,381 words in the passage answers and their term-level kappa agreement is 0.38 which is comparable to previous answer level agreements of 0.3 [11]. The average length of an answer passage is 45 words. Prior to the annotation phase, we manually filtered queries and selected the ones that are more likely to have passage-level answers. Among the 82 selected queries, only three had no annotated answers. This confirms that many web queries of this type can be answered using a small number of sentences.

4. EVALUATING ANSWER RETRIEVAL

As part of the passage retrieval task in HARD track, new evaluation measures were proposed. The proposed metrics, such as R-precision, consider relevant characters in the top retrieved passages [1]. The proposed character-level measures are generally similar to traditional document-level measures but use characters as opposed to documents. Similar character-level measures were used in the INEX ad hoc track evaluation and the TREC Genomics passage retrieval task [6, 5]. We use the Passage2 MAP measure from the TREC Genomics track for evaluating our systems. In this measure, each character is treated as a document and we calculate MAP measures based on the number of retrieved relevant characters. If a relevant character is retrieved more than once, we only consider its first retrieval as a hit. In addition to MAP, we also report character-based precision for the top 1 and 10 retrieved passages. Precision at 1 is particularly important due to the specific requirements of an answer retrieval system. In all of these evaluations, we assume that only passages marked "perfect" or "excellent" are relevant.

Table 1: Performance evaluation of QL and SDM for retrieving answers

Measure	MAP	P@1	P@10
QL	0.021	0.148	0.057
SDM	0.020	0.107	0.060
QL-Interpolated	0.022	0.073	0.062

5. PASSAGE RETRIEVAL METHODS FOR RETRIEVING ANSWERS

To study the effectiveness of existing passage retrieval methods for finding answers, we investigated different language model-based models. It has been shown previously that fixed-size window passages are the most effective for topical retrieval of documents in TREC corpora (e.g., [2]). Based on this observation and without further tuning, we fix the length of retrieved passages to 50 terms with an overlap of 25 terms. We use the two-phase passage retrieval feature in the Galago toolkit for all our experiments (www.lemur.org). Unless otherwise mentioned, we use default parameter values for existing methods such as smoothing parameters. We select the top 50 documents retrieved by the sequential dependence model (SDM) as input for our passage retrieval methods. Although we use default parameters, these values have been found to be effective in a range of previous passage and document retrieval experiments.

Query likelihood and SDM: In the first set of experiments, we employ existing language model techniques for retrieving passages. In these experiments, each passage is treated as a separate document and scored using the query likelihood model (QL) or the sequential dependence model (SDM) [9].

Further, we also experiment with an interpolation method in which we combine the score of the passage with the score of its document. We employ a homogeneity measure based on the length of the document to assign weights to each component. We use the length-based homogeneity function defined by Bendersky and Kurland that assign lower weights to document scores when document length increases [2].

Table 1 shows the performance evaluation of the language model methods. Interestingly there are no large differences between the three methods in terms of MAP and P@10. However query likelihood performs better than SDM in precision at high ranks. More interestingly, combining the document score with the passage score slightly increases the MAP and P@10 but decreases the performance at high ranks. The interpolation has a small effect mainly because our initial set of documents are all high quality documents and have very similar scores. It is worth noting that while the evaluation values are very low, they are comparable with previous studies on retrieving answers in TREC Genomics track [5].

Positional Model: In closely related work, Carmel *et. al.* used positional models (PM) for passage retrieval. They employ a $tf.idf$ scoring method for ranking passages:

$$score_{Psg}(p,q) = \sum_{t \in p \cap q} tf(t,p) \times idf(t) \qquad (1)$$

Table 2: Performance evaluation of PM for retrieving answers

Measure	MAP	P@1	P@10
PM-TFIDF	0.022	0.123	0.059
PM-Dirichlet	0.022	0.146	0.058
PM-SkewedGaussian	0.027	0.156	0.073

where tf is the query term pseudo-frequency in each passage that is estimated using positional model based on the distance between the passage and query term occurrences. The psuedo-frequency of a term t in a passage p is estimated as follows:

$$tf(t,p) = \sum_{pos \in occ(t,d)} \sum_{i=begin}^{i=end} kernel(pos, i) \qquad (2)$$

where $occ(t,d)$ is the set of all positions of t in the document, *begin* and *end* are the begin and end positions of the passage and $kernel(pos,i)$ is a kernel decay function that propagates term occurrence over all the positions in the document. In this experiment we use a Gaussian kernel function with $\sigma = 2000$, as it is shown to be an effective choice [4].

Beside the $tf.idf$ scoring model, we also use the estimated pseudo-frequencies in a query likelihood model using Dirichlet smoothing. The result for this method can be seen in table 2. As we can see, the methods have comparable performance to the other language model techniques while the query likelihood model performs slightly better than the $tf.idf$ method. Similar to our previous experiment, interpolating the document scores with the passage scores did not change our results in this experiment.

Non-symmetric Kernel Functions: In previous positional models for information retrieval, symmetric kernel functions have been employed that give the same propagation to positions before and after query term occurrences. After more in-depth investigation of the documents and queries in our collection, we found out that terms following a query term are more related to the query than terms before the query term. In order to model this property, we employ non-symmetric kernel functions that give higher value to positions after the query term occurrence. A distribution with such a kernel function is described as a distribution with positive skewness (skewness is the measure of asymmetry of a probability distribution that is zero for symmetric distributions). We employ a skew Gaussian kernel function that generalises Gaussian kernel to allow for non-zero skewness:

$$kernel_{skewed}(j,i) = e^{-\frac{(i-j)^2}{2\sigma^2}} \times \left[1 + \text{erf}(\frac{\alpha.(i-j)}{\sqrt{2}})\right] \qquad (3)$$

where *erf* is the error function and α is the skewness parameter which is set to one in our experiments. As in the previous experiment, we set the σ to 2000 and use the pseudo-frequencies in a query likelihood model. The results of this kernel function are shown in the last row of table 2. This method outperforms the others and confirms our intuition about non-symmetric kernel functions.

5.1 Query Expansion Methods in Answer Retrieval

The short length of passages makes it more likely to have a term mismatch problem between queries and answers. In other words, query terms may not occur frequently in an answer passage and other related terms will not contribute to the score. One of the possible solutions to this problem, that has been shown to be effective for document retrieval, is to expand the queries with related terms [7]. For example, the relevance model approach adds terms to the query that have high probabilities in the top retrieved documents. In this section, we study the effect of query expansion methods on the performance of answer passage retrieval. We employ relevance models (RM) to select and weight terms and we interpolate the selected terms with the original query terms, with a weight of 0.85 for original terms and 0.15 for expansion terms [7]. Without further tuning, we set the number of feedback documents and feedback terms to 25. We explore two options for selecting terms. In the first approach, we select terms from the top retrieved documents and in the second approach we select terms from the top retrieved passages. We use the expanded queries in the positional model using skewed Gaussian kernel.

Table 3: Effect of Query Expansion Methods

Measure	MAP	P@1	P@10
RM on documents	0.027	0.133	0.071
RM on passages	0.027	0.153	0.067

Table 4: Effect of input document

Measure	MAP	P@1	P@10
Top 5	0.008	0.121	0.043
Top 10	0.012	0.136	0.055
Top 25	0.023	0.169	0.074
Only Relevant	0.099	0.302	0.179

Table 3 shows the performance results of the two methods. Compared to document retrieval where query expansion can generally improve performance, neither of the two approaches outperforms their non-expanded counterparts. Even more surprisingly, document-based expansion decreases the performance, especially in the top ranks.

5.2 Quality of Initial Retrieval

In all our experiments so far, we used the top 50 retrieved documents from SDM as the input to our passage retrieval methods. In this section, we investigate if using fewer documents, presumably with higher quality, can affect the performance of our passage retrieval methods. To this end, we repeat the previous experiment with the top 5, 10 and 25 documents that are retrieved by SDM. The positional model using the skewed Gaussian kernel is the retrieval method in this experiment. The top three rows in table 4 show the results. Unsurprisingly, the MAP measure decreases when we use very few documents. However, we can see that precision at 1 and 10 is also decreased when we use 5 or 10 documents. The best result is obtained when we use the top 25 documents and precision at 1 has the most improvement. This shows that passage retrieval methods are sensitive to the quality of input documents and the top retrieved documents do not necessarily have highest quality.

Given that the document retrieval phase can have a significant effect on the passage retrieval phase, it is worth studying the upper-bound case where we have a perfect document retrieval method that returns only relevant documents. To this end, we use only our annotated documents as the input and re-run our our passage retrieval methods. The last row

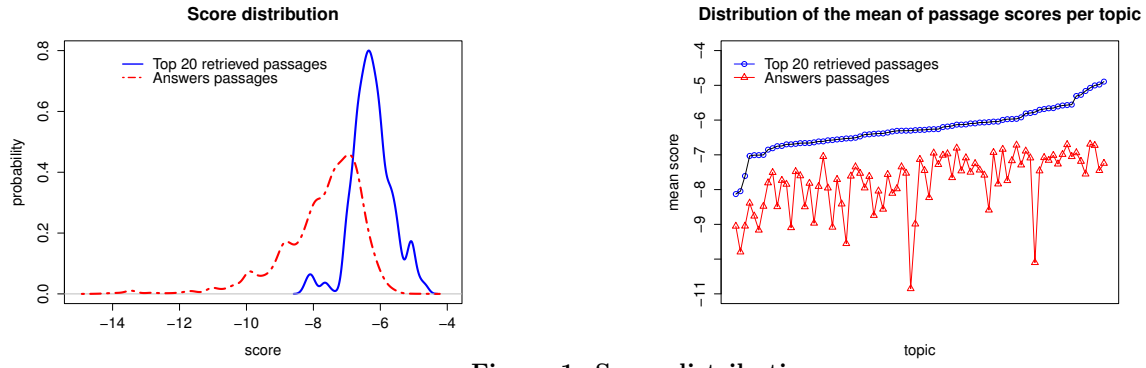

Figure 1: Score distribution

in table 4 shows the result of the experiment. As expected, all the measures are significantly improved when we retrieve passages from only relevant documents. However, while 30% of retrieved characters are relevant characters, our passage retrieval methods are still not very effective for finding the answers in the documents.

5.3 Discussion and Future Directions

Our experiments show that current passage retrieval methods that focus on topical relevance fail to perform well for answer passage retrieval. In more in-depth analysis, we looked into the scores that are assigned by our retrieval method to the annotated answers and compared them to the scores of the top 20 retrieved passages. Figure 1 shows the distribution of scores over all the topics on the left and the mean of the scores per topic on the right. Scores are the log-scale values assigned by the language model. As we can see, the answer passages have distinctively lower scores compared to the top 20 retrieved passages and this is consistent across all the topics. Interestingly there are some answers, mostly with scores less than -11, which do not contain any of the query terms.

The results show that features based on term frequencies are not sufficient for retrieving answer passages. Although the results could be improved somewhat by more intensive tuning, we believe that there is an obvious need to incorporate other types of features into retrieval models. We are currently exploring a range of features, including linguistic features studied for CQA data [12] and features used in generating summaries [10].

6. ACKNOWLEDGMENTS

This work was supported in part by IBM subcontract 4913003298 under DARPA prime contract HR001-12-C-0015, and by the Australian Research Council (DP140102655). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

7. REFERENCES

[1] J. Allan. HARD track overview in TREC 2004 - high accuracy retrieval from documents. In *Proceedings of TREC*, 2004.

[2] M. Bendersky and O. Kurland. Utilizing passage-based language models for document retrieval. In *Proceedings of ECIR'08*, pages 162–174, 2008.

[3] J. Callan. Passage-level evidence in document retrieval. In *Proceedings of SIGIR'94*, pages 302–310, 1994.

[4] D. Carmel, A. Shtok, and O. Kurland. Position-based contextualization for passage retrieval. In *Proceedings of CIKM'13*, pages 1241–1244. ACM, 2013.

[5] W. R. Hersh, A. M. Cohen, P. M. Roberts, and H. K. Rekapalli. TREC 2006 genomics track overview. In *Proceedings of TREC'06*, 2006.

[6] J. Kamps, J. Pehcevski, G. Kazai, M. Lalmas, and S. Robertson. INEX 2007 evaluation measures. In *proceedings of INEX workshop*, pages 24–33, 2007.

[7] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of ACM SIGIR'01*, pages 120–127. ACM, 2001.

[8] X. Liu and W. Croft. Passage retrieval based on language models. In *Proceedings of CIKM*, pages 375–382, 2002.

[9] D. Metzler and W. Croft. A Markov random field model for term dependencies. In *Proceedings of SIGIR*, pages 472–479, 2005.

[10] D. Metzler and T. Kanungo. Machine learned sentence selection strategies for query-biased summarization. In *proceedings of SIGIR workshop on Learning to Rank for Information Retrieval*. ACM, 2008.

[11] T. Sakai, D. Ishikawa, N. Kando, Y. Seki, K. Kuriyama, and C.-Y. Lin. Using graded-relevance metrics for evaluating community QA answer selection. In *Proceedings of WSDM'11*, pages 187–196, 2011.

[12] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

[13] D. Zhang and W. Lee. A language modeling approach to passage question answering. In *Proceedings of TREC*, pages 489–495, 2003.